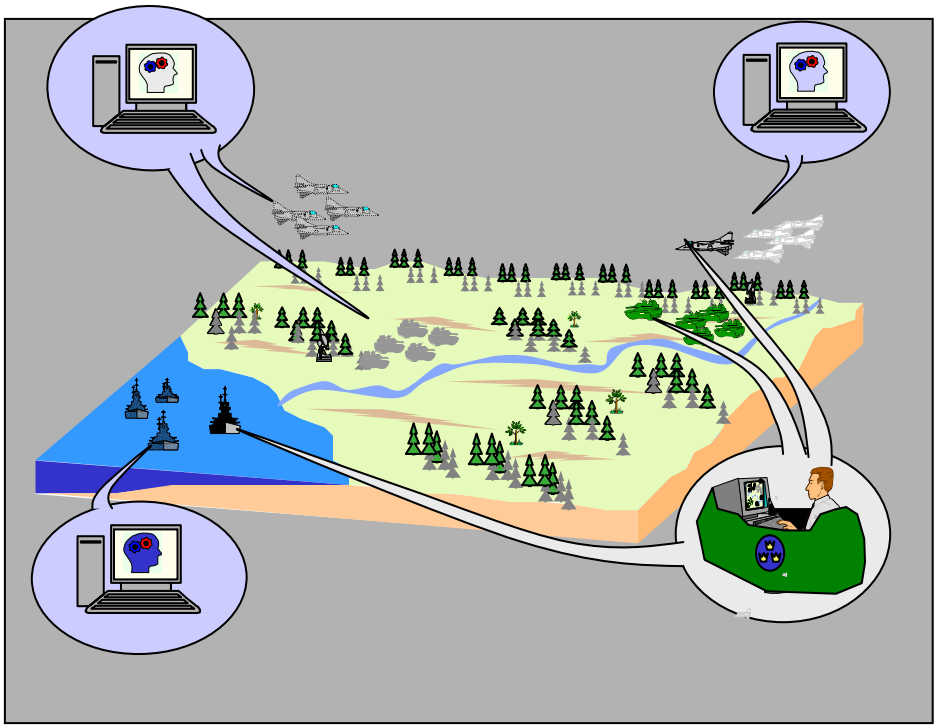
FOI
TOTALFÖRSVARETS
FORSKNINGSINSTITUT

FOI-R--0688--SE
December 2002
ISSN 1650-1942
Användarrapport

Martin Castor, Niklas Wallin, Sten-Åke Nilsson, Farshad Moradi

Datorgenererade styrkor - Metoder och möjligheter



Ledningssystem
581 11 LINKÖPING

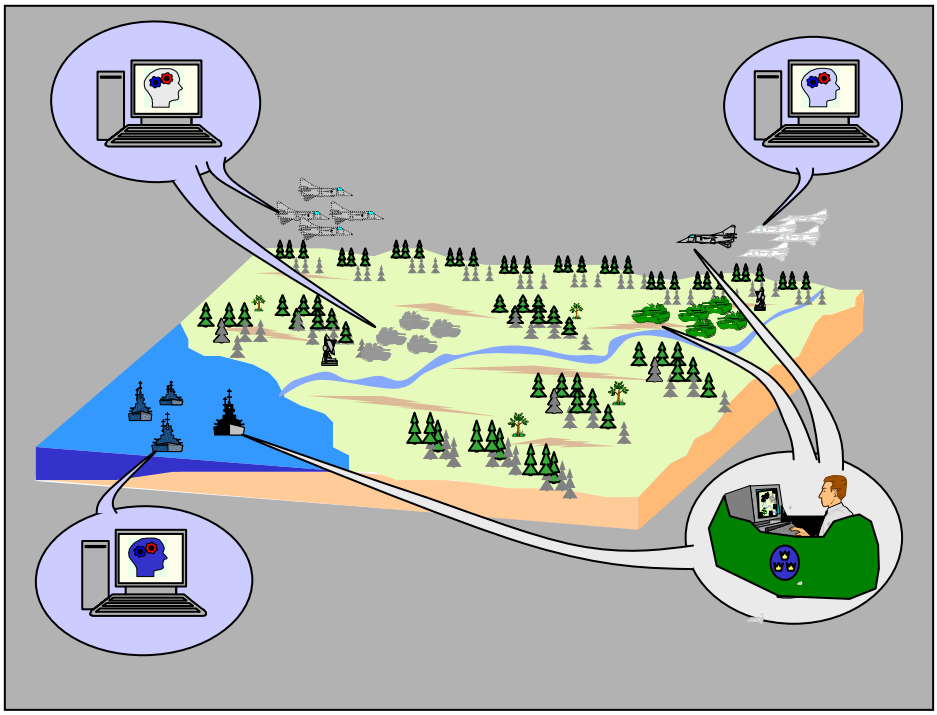
Systemteknik
172 90 STOCKHOLM

TOTALFÖRSVARETS FORSKNINGSINSTITUT
Systemteknik
172 90 Stockholm

FOI-R--0688--SE
December 2002
ISSN 1650-1942
Användarrapport

Martin Castor, Niklas Wallin, Sten-Åke Nilsson, Farshad Moradi

Datorgenererade styrkor - Metoder och möjligheter



Utgivare	Rapportnummer, ISRN	Klassificering
Totalförsvarets Forskningsinstitut - FOI	FOI-R--0688--SE	Användarrapport
Systemteknik	**Forskningsområde**	
172 90 Stockholm	2. Operationsanalys, modellering och simulering	
	Månad, år	**Projektnummer**
	December 2002	E6029
	Verksamhetsgren	
	5. Uppdragsfinansierad verksamhet	
	Delområde	
	21 Modellering och simulering	

Författare/redaktör	Projektledare
Martin Castor	Sten-Åke Nilsson
Niklas Wallin	**Godkänd av**
Sten-Åke Nilsson	Monica Dahlén
Farshad Moradi	**Uppdragsgivare/kundbeteckning**
	FM
	Tekniskt och/eller vetenskapligt ansvarig

Rapportens titel

Datorgenererade styrkor - Metoder och möjligheter

Sammanfattning

Modellering och simulering (M&S) är ett kraftfullt verktyg som används för att stödja utveckling av framtida militära koncept och system såväl som till att förbättra träning och användning av befintliga styrkor och utrustningar, samt att i förväg analysera militära operationer i nya miljöer.

Det finns ett antal teknologier som har stor betydelse för resultatet när man utnyttjar simuleringar för militära ändamål. Datorgenererade styrkor (Computer Generated Forces, CGF) är en sådan teknologi. CGF erbjuder representationer av militära enheter på olika aggregationsnivåer, och användes alltmer för att befolka distribuerade simuleringar.

Försvarsmakten har behov av mer realistiska modeller av mänskliga operatörer i sina simuleringar. FOI har sedan 2001 i uppdrag att genomföra studier inom området. Syftet med projektet är att skapa en grund för utveckling av CGF och att bygga upp ett komponentbaserat bibliotek med representativa kognitiva och beteendevetenskapliga modeller av operatörer vilka kan användas i olika simuleringsmiljöer.

Denna rapport är en beskrivning av det arbete som utfördes under 2001 och 2002. Arbetet handlade dels om att inventera marknaden vad gäller arkitekturer och verktyg och att utvärdera ett par olika arkitekturer genom att utveckla enkla modeller av flygförare. Under 2002 har arbetet också fortsatt med utveckling av flera prototyper, design av struktur och gränssnitt hos modellkomponenter samt en påbörjad uppbyggnad av komponentbaserade bibliotek. Slutsatser och projektets planer för fortsatt arbete redovisas också i rapporten.

Nyckelord

M&S, CGF, datorgenerade styrkor, simulering, kognitiv modellering, HBR, arkitekturer, MSI

Övriga bibliografiska uppgifter	Språk Svenska
ISSN 1650-1942	**Antal sidor:** 77 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Issuing organization
FOI – Swedish Defence Research Agency
Systems Technology
SE-172 90 Stockholm

Report number, ISRN
FOI-R—0688--SE

Report type
User report

Research area code
2. Operational Research, Modelling and Simulation

Month year
December 2002

Project no.
E6029

Customers code
5. Commissioned Research

Sub area code
21 Modelling and Simulation

Authors
Martin Castor
Niklas Wallin
Sten-Åke Nilsson
Farshad Moradi

Project manager
Sten-Åke Nilsson

Approved by
Monica Dahlén

Sponsoring agency
Swedish Armed Forces

Scientifically and technically responsible

Report title (In translation)
Computer Generated Forces - Methods and Means

Abstract
Modelling and Simulation, M&S, is a powerful tool that is used to support development of military concepts as well as training, studies and analysis of military operations in different environments.
There are a number of important technologies that can be applied to military simulations. Computer Generated Forces, CGF, is one such technology. CGF are used as Human Behavior Representations, HBR, of individuals or groups as operators or commanders in military simulations.
Defence driven M&S is demanding more realistic HBR models in simulations. To confront the growing need of such models, the Swedish Armed Forces has sponsored a study of CGF development since 2001, conducted at FOI. The main purpose of the project is to construct a framework to support CGF development and to maintain a component based library of HBR models to be used in existing and future simulations.
This report describes the following activities performed during 2001 through 2002:

- Evaluations of existing architectures and tools to produce HBR models
- Five prototype descriptions
- Proposals for interface design to produce “plug’n’play” compatibility with different simulation systems
- Proposals for a consistent praxis of developing CGF
- Some words of how to incorporate experimental data into HBR models
- VV&A considerations
- Future work

Keywords
M&S, CGF, Computer generated forces, simulation, cognitive modelling, architectures, human factors

Further bibliographic information

Language Swedish

ISSN 1650-1942

Pages 77 p

Price acc. To pricelist

Sammanfattning

Modellering och simulering (M&S) är ett kraftfullt verktyg som används för att stödja utveckling av framtida militära koncept och system. Modellering och simulering används också till att förbättra träning och användning av befintliga styrkor och utrustningar, samt att i förväg analysera militära operationer i nya miljöer.

Det finns ett antal teknologier som har stor betydelse för resultatet när man utnyttjar simuleringar för militära ändamål. Datorgenererade styrkor (Computer Generated Forces, CGF) är en sådan teknologi. Datorgenererade styrkor som används i simuleringar erbjuder representationer av militära enheter på olika aggregationsnivåer, och användes alltmer för att befolka distribuerade simuleringar.

Vid FOI, Institutionen för Systemmodellering startade under 2001 ett förstudieprojekt vid namnet ”Datorgenererade Styrkor, metoder och möjligheter”. Projektet, som utfördes på uppdrag från FM, utgick ifrån FOA:s mångåriga verksamhet och erfarenhet inom beteendevetenskap, distribuerad simulering och konstruktion av komplexa modeller samt Försvarsmaktens behov vad gäller modellering av mänskligt beteende och datorgenererade styrkor. Syftet med projektet har framförallt varit att kartlägga CGF-området, identifiera befintliga metoder och tekniker, identifiera möjligheter som området kan erbjuda, i begränsad omfattning prova och jämföra tekniker, samt att utveckla riktlinjer för hur bibliotek med modeller av agenters kognition och beslutsfattande kan byggas upp.

Rapporten är en beskrivning av området och det arbete som utfördes under 2001 och 2002. Arbetet handlade dels om att inventera marknaden vad gäller arkitekturer och verktyg och göra jämförelser mellan dessa, samt utvärdera ett par olika arkitekturer genom att utveckla enkla modeller av flygförare. Under 2002 har arbetet också fortsatt med utveckling av flera prototyper, design av struktur och gränssnitt hos modellkomponenter samt hur man kan bygga komponentbaserade bibliotek. Slutsatser från detta arbete och projektets planer för fortsatt arbete redovisas också i rapporten.

Innehållsförteckning

Förteckning över figurer och tabeller

Förkortningar

ACT-R	Adaptive Control of Thought Rules
ACT-PM	Adaptive Control of Thought Perceptual Motor
AI	Artificiell Intelligens
AMBR	Agent-based Modeling and Behavioral Representation
ANN	Artificiella Neurala Nät
CGF	Computer Generated Forces
CHRIS	(projekt med avsikt att utveckla en standard för ”human behavior representation)
COGNET	COGnition as a NEtwork of Tasks
CTA	Cognitive Task Analysis
DARPA	Defence Advanced Research Projects Agency
D-COG	Distributed COGniton
D-OMAR	Distributed Operator Model Architecture
DIS	Distribuerad Interaktiv Simulering
DMSO	Defence Modeling and Simulation Office.
EPIC	Executive-Process Interactive Control
FLAMES	FLexible Analysis and Mission Effectiveness System
FLSC	Flygvapnets LuftstridsSimuleringCenter
FOM	Federation Object Model
GOMS	Goals, Operators Methods and Selection Rules
HBR	Human Behavior Representations
HLA	High Level Architecture
HTA	Hierarchical Task Analysis
HOBM	Human and Organizational Modeling
M&S	Modellering och Simulering
MIDAS	Man-machine Integrated Design and Analysis System
Mod-SAF	Modular Semi-Automated Forces
MSI	Människa System Interaktion
NGOMSL	”Natural” GOMS Language
PMF´s	Performace Moderator Functions
RTI	Run Time Infrastructure
SAF	Semi-Automated Forces
SEDRIS	Synthetic Environment Data Representation and Interchange Specification
Soar	State, operator and result
Soar/IFor	State, operator and result/Intelligent Forces
SOM	Simulation Object Model
STAGE	Scenario Toolkit And Generation Environment
STRIVE	Synthetic Tactical Real-time Interactive Virtual Environment
TACSI	TACtical SImulation
UTA/UTB	Utbildningsanalys hjälpmedel/anläggning
VV&A	Verifiering, Validering och Ackreditering

1. Projektet

Vid FOI, Institutionen för Systemmodellering startade under 2001 ett förstudieprojekt vid namnet ”Datorgenererade Styrkor, metoder och möjligheter”. Projektet, som utfördes på uppdrag från FM utgick ifrån FOA:s mångåriga verksamhet och erfarenhet inom beteendevetenskap, distribuerad simulering och konstruktion av komplexa modeller samt Försvarsmaktens behov vad gäller modellering av mänskligt beteende och datorgenererade styrkor.

Datorgenererade styrkor (Computer Generated Forces, CGF) är en term som används för att beskriva modeller och simuleringar som erbjuder representationer av militära styrkor, och har traditionellt använts för att befolka simuleringar med objekt och aktörer för att skapa en realistisk miljö.

Rapporten beskriver det utförda arbetet, identifierade frågor och framtagna modeller samt redovisar projektets slutsatser från detta arbete. I kapitel 1 beskrivs bakgrunden till projektet. Kapitel 2 redovisar en kartläggning och kriterier för jämförelse av olika arkitekturer. I kapitel 3 diskuteras varför kognitiv modellering är viktigt för CGF-utveckling. Kapitel 4 är en beskrivning av arbetet med de prototyper som utvecklats hittills. I kapitel 5 beskrivs några nationella och internationella projekt. Rapporten avslutas med slutsatser och förslag på fortsatt verksamhet i kapitel 6.

Begrepp och forskningsområden som i stor utsträckning är kopplade till CGF:er är ”Human and Organizational Modeling” (HOBM), ”Human Behavior Representation” (HBR), artificiell intelligens (AI), kognitiv modellering, och modellering och simulering (MoS).

1.1. Behov

För att simuleringar skall vara användbara måste de avbilda verkligheten på ett för användningen korrekt sätt. I träningssimulatorer är det sålunda av största vikt att modellerna är realistiska, dynamiska och kan reagera på ett verklighetstroget sätt gentemot situationen och omgivningen. Om detta ej är fallet kan ett felaktigt beteende tränas hos de människor som deltar i simuleringen. Ofta krävs många sam- och motverkande objekt i en och samma övning för att kunna skapa en realistisk övningsmiljö och det är sällan praktiskt möjligt att låta bemannade simulatorer representera alla dessa enheter.

Datorgenererade styrkor (“Computer Generated Forces”, CGF:er) är därför ett grundläggande byggblock i teknologin för Distribuerad Interaktiv Simulering (DIS) eftersom de erbjuder den enda ekonomiskt effektiva metoden som finns för att befolka DIS-miljön med tillräckligt många aktörer. I takt med att antalet simulatorer och simuleringsmiljöer i Sverige ökar kommer också behovet av att kunna befolka dem att öka. Exempel på några simulatoranläggningar som inom en snar framtid kommer behöva intelligenta datorgenererade styrkor är FLSC (Flygvapnets LuftstridsSimulering-Center) och marinens ledningsträningsanläggning MIMER. Andra avancerade simuleringsmiljöer t.ex. SMART-Lab kommer också att vara betjänta av datorgenererade styrkor med ett självständigt och realistiskt beteende.

Uppkomsten av CGF och SAF (Semi-Automated Forces) -simuleringar skedde genom DARPA:s (Defence Advanced Research Projects Agency) försök med SIMNET-programmet under den senare hälften av 80-talet. SIMNET bestod av en serie stridsvagnsimulatorer som var sammankopplade via nätverk och som använde ett gemensamt kommunikationsprotokoll (SIMNET-protokollet) för att dela på statusinformation. I början av projektet räckte det med de sammankopplade simulatorerna för att träna små grupper. Det blev dock snart klart att för att simulera större taktiska övningar med tillräckligt många aktörer behövdes en mer skalbar metod än att enbart koppla till fler simulatorer. För att tillgodose detta behov, utvecklade man datorprogram som möjliggjorde att mänskliga simulator-operatörer kunde skapa ett antal simulerade objekt, som t.ex. stridsvagnar eller flygplan, och kontrollera deras beteende.

Sedan dess har utvecklingen inom området fortsatt och CGF-programmen har ökat i antal och komplexitet. Det mest kända exemplet är förmodligen Mod-SAF (Modular Semi-Automated Forces) som under 90-talet växte från att vara ett enkelt system för att försörja simuleringar med markbaserade plattformar till ett komplext simuleringssystem med förmåga att simulera stora mängder av mark-, luft- och sjöplattformar, samt andra omgivningsobjekt. Detta har resulterat i att programmet har växt från 100 000 rader C-kod till ca 1000 000 rader i den senaste versionen. Det finns också andra CGF och CGF-liknande simuleringar som har utvecklats för olika syften. Trots de omfattande investeringarna som gjorts i dagens CGF-tillämpningar har t.ex. ModSAF en relativt enkel kunskapsrepresentation, och simuleringarna blir inte tillräckligt detaljerade och realistiska. Inom det amerikanska försvaret verkar trenden nu gå mot kognitiv modellering och agentteknologi när man skall utveckla CGF tillämpningar.

Datorgenererade styrkor och datorbaserade modeller av operatörers och beslutsfattares kognition kommer i framtiden att användas för ett antal olika syften, bland annat:

- Befolka den simulerade världen (för träning).
- Befolka den simulerade världen (för taktikutveckling).
- SBA (simulation based acquisition) i design frågor (t.ex. operativa konsekvenser av ett hjälmsikte)
- Utformning av beslutsstödsystem.
- Inledande faser i taktikutveckling med enbart automatiserade styrkor.
- Taktikstöd till beslutsfattare i realtid.

1.2. Mål

Det långsiktiga och övergripande målet med projektet är att studera möjligheter och förutsättningar för CGF-utveckling, samt att bidra till att bygga upp ett försvarsmaktsgemensamt referensbibliotek med modeller av datorgenererade styrkor på olika nivåer. Samtidigt finns verksamhetsmålet att skapa gedigen kunskap inom området datorgenererade styrkor för att bl. a. kunna stödja modellering och simuleringsverksamheten inom Försvarsmakten, FOI och industrin.

Delmål på kortare sikt, varav några studerats närmare hittills, är:

- Att inventera och beskriva de kognitiva modelleringsarkitekturer, AI-tekniker för kunskapsrepresentation och CGF-programvaror som finns tillgängliga. Det finns ett stort antal arkitekturer för kognitiv modellering, artificiell intelligens, kunskapsrepresentation och CGF-utveckling på marknaden och i forskningslitteraturen. De har alla sina fördelar och tillkortakommanden. En del av förstudiens arbete har gått ut på att inventera marknaden, definiera kriterier för utvärdering, och välja någon ansats som är lämplig att utveckla demonstratorer i.

- Att undersöka möjligheterna med att få in experimentellt grundade beslutsmodeller i CGF-programmen. De syntetiska entiteternas beteende bör efterlikna det beteende som de mänskliga operatörerna uppvisar. Ett sätt för att åstadkomma detta är att definiera beteendet hos modeller med hjälp av experimentella data från verkliga operatörer. Data på hur dessa operatörer uppträder måste samlas in experimentellt i fältstudier vilket är ett mycket omständligt arbete. På Institutionen för Människa-System-Interaktion vid avdelningen för Ledningssystem har man samlat in data från olika domäner, t.ex. flygförare. Ett delmål i projektet har därför varit att undersöka hur man kan infoga dessa experimentellt grundade beslutsmodeller i CGF-programmen. Agenter där dessa beslutsmodeller infogats skall senare i projektet utvärderas i träningsmiljöer och simulatorer som t.ex. FLSC eller FENIX (FENIX är en efterföljare till simulatorn FOSIM och utvecklas av avdelningen för Systemteknik).

- Att utreda möjligheterna för att CGF-programmen/de kognitiva modellerna kan styra objekt i de scenariogenereringsprogram som används i Försvarsmaktens simulatoranläggningar eller simulationer. Beroende på vilken modelleringsarkitektur eller programvara som används så

kommer det vara olika lätt att koppla detta till scenariogeneratorer som t.ex. STAGE, FLAMES eller TACSI. Projektet skall också undersöka möjligheterna med att innesluta beteendemodellen eller den kognitiva modellen i ett HLA kompatibelt skal så att modellen kan få indata från omvärlden och ge styrorder ut genom skalet, oavsett hur modellen implementerats innanför skalet. På så sätt skulle man enklare kunna koppla dessa modeller till simulatorer och simuleringar som t.ex. FLSC, FENIX och MIMER.

- Att utveckla metodik för validering och verifiering av kognitiva modeller. Hur skall verifiering och validering av kognitiva modeller gå till? Det är en sak att utvärdera en agents beteende utifrån vad som är korrekt beteende enligt gällande doktrin, men det blir svårare att genomföra verifiering och validering om man använder sig av deskriptiva kognitiva modeller av det faktiska beteendet, d.v.s. modeller som ibland tar fel beslut p.g.a för hög mental arbetsbelastning. Idag finns det inga direkta lösningar för dessa problem. Ett framtida mål i projektet bör vara att utveckla metodik för verifiering och validering av kognitiva modeller.

1.3. Flera användningsområden för CGF och kognitiv modellering

Det har sedan början av 90-talet förekommit försök att modellera förband med ett så intelligent beteende, att det inte fordras någon mänsklig operatör alls för att övervaka dem. Det mest kända av dessa tidiga försök var det s.k. Soar/IFOR-projektet (Soar, ”State, Operator And Result”/Intelligent Forces), där man försökte skapa modeller av piloter som skall kunna fungera i alla normala situationer som inträffar under flyguppdrag. Man utvecklade också modeller av flygstridsledare som kommunicerar med de flygande pilotmodellerna under uppdragen. Dessa modeller har använts i ett antal övningar under åren. Bland annat kan man nämna de amerikanska STOW-E, 1994, samt Coyote- och Roadrunner-övningarna, 1998, som var flera dygn långa, fullständiga simulationer av luftkrig med hundratals agenter och bemannande simulatorer.

Syftet med dessa CGF:er har framförallt varit att skapa större, mer realistiska och meningsfulla simuleringar för taktiska övningar genom att simulera egna och motståndarförband.
Datorgenererade förband kan också med fördel användas i förhandsanalys av planerade militära operationer, taktikutveckling, materiellanskaffningsprocesser och utveckling av nya system. Dessa olika användningsområden ställer olika krav på de datorgenererade styrkorna och på modellerna som styr deras beteende. Om man t.ex. i en materielanskaffningsprocess vill studera det operativa utfallet av införandet av ett hjälmsikte behöver man ha mer detaljerade modeller av flygförarnas visuella perception än om de modellerade flygförarna ”bara” skall vara motståndare i en BVR-strid.

Den nya krigföringen handlar i stor utsträckning om informationsövertag och att utifrån det kunna förutse motståndarens handlande. I och med detta kommer detaljerad modellering av beslutsfattandet och kognitionen hos fienden att behövas för att informationsstriden skall kunna modelleras. Det kommer inte att räcka med enkla modeller av simulerade beslutsfattare, utan striden måste kunna simuleras på många olika aggregationsnivåer. Det kommer inte räcka med en utnötningsalgoritm för att avgöra striden, eftersom inte sidan med mest eldkraft nödvändigtvis vinner. Asymmetrisk krigföring kommer att höra till vardagen och för att kunna förutspå utfallet här måste man kunna modellera på individnivå. Det är också först då man kan få fungerande prediktiva analyser av hur informationskriget fungerar samt inverkan av kulturella och sociologiska effekter.

1.4. Nytta för FM

Genom att använda datorgenererade styrkor kan man minimera både kostnaden för organisation, ledning och dirigering av enheter som behövs för en effektiv och meningsfull simulering. Det skulle ur försvarets synvinkel vara värdefullt med att ha ett bibliotek av generiska styrkor som representerar olika militära och icke-militära enheter på olika aggregationsnivåer. Dessa skulle kunna användas som ”plug’n’play” moduler, som via ett givet gränssnitt kan kopplas till de simuleringar där de behövs. Användningsområdena är många, t.ex. träning, utbildning och analys.

Området ”Human Behavior Representation” är tydligt utpekat som ett centralt område i DMSO:s Master Plan (Objective 4 i DMSO, 1995) och har blivit så omfattande att det årligen arrangeras speciella konferenser om Computer Generated Forces and Behavioral Representation i USA.
Följaktligen är det av största vikt att Sverige besitter kompetens inom detta område, om inte annat för att datorgenererade styrkor är svåra att komma åt inom den kommersiella marknaden, även om grundarkitekturer kan vara gratis. Om man t ex betraktar en applikation som Tac-Air-Soar, som innehåller 7000 regler för hur AI-kontrollerade flygplan skall agera, är det uppenbart att sannolikheten för att man skall få tillåtelse att titta på koden är väldigt liten. Detta tyder på att det mesta inom detta område måste byggas upp inom landet och vi behöver ett försprång innan behoven blir akuta.
Inom den närmaste framtiden kommer det att finnas ett behov av sådana modeller i bl. a FLSC, FENIX, MIMER.

Andra intressanta kommande militära forskningsprojekt som skulle vara betjänta av CGF:er för att simulera militära och icke-militära enheter på olika nivåer är LedsystX, Nätverkstrid, Strid i bebyggelse och IT-Krig/Framtida ledningssystem.

1.5. Genomförande

Projektgruppen har under 2001 och 2002 bestått av sammanlagt fyra forskare (tre från avdelningen för Systemteknik och en från avdelningen för Människa-System-Interaktion, numera ingående i avdelningen för Ledningssystem) samt en examensarbetare vardera år. Följande aktiviteter har genomförts:

- Inledande arbete med kartläggning av området genom inventering av CGF- och kognitiva modelleringsprojekt under de senaste åren. Kunskapsuppbyggnad har bl.a. skett genom deltagande i en doktorandkurs i kognitiv modellering given av LiTH samt NATO RTO:s ”SAS Simulation Series” på avdelningen för Försvarsanalys. Arbetet har inkluderat diskussioner, deltagande i presentationer och nära samarbete med FLSC, FMV, Pitch AB och Soartech i USA.

  Därefter har projektet följt och bevakat utvecklingen för området inom- och utomlands. Detta har bl. a. inneburit deltagande i både den tionde och elfte konferensen ”on Computer Generated Forces and Behavioral Representation” i USA. Dessutom har projektet upprättat ett utbyte med bl.a doktorander från University of Michigan och Soartech i USA samt initierat ett samarbete med ”National University of Singapore” som kan påbörjas under 2003.

- Arbete med utveckling av datorgenererade piloter med hjälp av experimentellt grundade beteendevetenskapliga data. Projektet har studerat hur modeller, av hur psykologiska och psykofysiologiska faktorer påverkar en flygförares beteende, kan infogas in i agentprototyper. Modeller för interaktionen mellan mental arbetsbelastning, situationsmedvetande och operativ prestation och experimentella data har tillhandahållits av MSI projektet Mental Arbetsbelastning och Prestation. I samband med detta har projektet skapat uppgiftsanalyser efter bl.a. samarbete med Soartech och flygförare på FLSC och F17.

- En teoretisk kartläggning av ett stort antal modelleringsarkitekturer och ansatser och utveckling av kriterier för jämförelse har skett och presenteras i avsnitt 2 och appendix A. En jämförelse av att arbeta med arkitekturerna Soar och Agent Factory presenteras i avsnitt 4.

- Förslag på bibliotheksstruktur har utvecklats. Som ett led i detta har projektet tagit fram gränssnitt för att skapa ”plug’n’play” kompatibla modeller som enkelt ska kunna stoppas in i olika simuleringar. Strukturen stödjer också ett standardiserat sätt för utveckling av modeller.

- Ett gränssnitt, ”Agentinterface”, till en av de mest använda arkitekturerna, Soar, har tagits fram. Gränssnittet är ett led i bibliotheksstrukturen och gör det lättare för en utvecklare att utnyttja kognitiva arkitekturer. Ett liknande gränssnitt för HLA kompatibilitet har påbörjats.

- Flera CGF prototyper har utvecklats enligt bibliotheksstrukturen och med hjälp av bl.a ”Agentinterface”. Prototyperna har utnyttjat modelleringsarkitekturer som Soar och Agent Factory och har bl.a. befolkat Fenix, Flames, Quake2, och Acm (se appendix B).

- Ett samarbete har initierats med VV&A-projektet på FOI för att kunna utreda/utveckla metodik för validering och verifiering av kognitiva modeller. Projektet har i samråd med VV&A-projektet tagit fram en konceptuell modell för de utvecklade agenterna som ett första led i VV&A-arbetet.

- Projektet har internt FOI upprättat en nätsida för projektet, där man kan få aktuell information om projektet och följa pågående verksamhet.

2. Värdering av arkitekturer och ansatser för CGF-implementation

När man skall utveckla CGF:er finns det en mängd olika ansatser och arkitekturer som använts i forskning och utveckling de senaste 20 åren. Olika behov, tillämpningar och begränsningar har gjort att ett flertal olika sätt att lösa problemet existerar. Dessa olika arkitekturer har olika egenskaper, möjligheter och begränsningar och valet av arkitektur bör basera sig på den modelleringsuppgift som skall lösas. I detta avsnitt beskrivs ett antal kriterier som bör beaktas vid val av modelleringsansats. Dessutom finns en preliminär kartläggning av existerande modelleringsansatser samt värdering enligt angivna kriterier.

2.1. Värderingskriterier för jämförelsen av arkitekturer

De ansatser och arkitekturer som granskats av projektgruppen värderas med hjälp av punkterna i matrisen nedan. Nedan angivna kriterier återfinns delvis i McClanahan mfl (2001), men är kraftigt omarbetade och utvecklade.

Namn	Arkitekturens namn
Projektets konfidens i bedömningen	Projektmedlemmarna inom projektet har under 2001-2002 byggt upp sin kunskap om de modelleringsansatser som används idag och har nått olika långt i sin förståelse av de olika ansatserna. Projektdeltagarnas konfidens i sin bedömning av arkitekturen/ansatsen anges i detta fält i tre steg: • Hög: någon av projektdeltagarna har tittat närmare på arkitekturen rent handgripligt och besitter praktisk kunskap om modelleringsarbete i beskriven arkitektur. • Medel: projektet besitter relativt ingående teoretisk kunskap om arkitekturen. • Låg: projektdeltagarna besitter bara översiktlig kunskap, ofta baserat på utvecklarnas eget försäljningsmaterial.
Vetenskaplig grund för att modellera:	De olika ansatserna har olika fokus och användningsområde. Vad som bedöms vara onödigt komplext att modellera i en arkitektur är i fokus för en annan. Modellering i de olika arkitekturerna i olika aspekter är därför vetenskapligt underbyggd i olika stor utsträckning.
Perceptuellt och motoriskt beteende	Modellering av perceptionen (d.v.s. informationsinhämtning med olika sinnen som syn eller hörsel) och modellering av motoriskt beteende (t.ex. styra en styrspak eller trycka på knappar) kan tyckas vara två relativt skilda saker men i beteendevetenskapliga beskrivningar behandlas de ofta tillsammans.
Kognitivt beteende	I detta fält anges hur väl arkitekturen lämpar sig för att modellera beslutsfattande och problemlösningsförmåga. I många arkitekturer anses detta vara det viktigaste.
Grupp beteende	I detta fält anges om arkitekturen har något särskilt stöd för att modellera gruppers eller aggregerade enheters uppträdande. Finns det t.ex. stöd för att modellera en bataljon som en aggregerad enhet med intern kommunikation osv.

Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	I detta fält anges hur användbar arkitekturen eller ansatsen är för att modellera stora och komplexa problem av den typ verkligheten erbjuder, d.v.s. inte bara problemlösningsförmåga i en begränsad mikrovärld. Här anges också om det arkitekturen är anpassad till någon särskild nisch eller inte, d.v.s. om det finns inbyggda antaganden om vad modellerna skall kunna resonera om eller inte.
”Situatedness”	I vilket grad hanterar arkitekturen begreppet situatedness? ”Situatedness” kan översättas med ”Att vara i världen,” d.v.s. att agenten sitter i ett flygplan och får information genom sina displayer och agerar genom att trycka på knappar. Modelleras t.ex. flygplanet och agenten separat eller inte?
Deterministiskt system	Blir det alltid samma utfall (givet samma situation) eller inte? Ibland är detta en önskvärd egenskap, t.ex. vid taktikutveckling (eftersom man vill kunna upprepa resultaten i en stor mängd simuleringar) och ibland inte t.ex. vid träning (eftersom de mänskliga operatörerna då snabbt lär sig agenternas beteende).
Kunskapsrepresentation och beteenderepresentation	Hur beskrivs de syntetiska entiternas kunskap och beteenden? En central fråga som i stor utsträckning påverkar modellerna problemlösningsförmåga och utvecklingskostnad.
Inlärningsförmåga eller träningsbehov	Har agenterna förmåga att utveckla ny kunskap och automatiskt utveckla nya beteenden alternativt behöver de tränas (se t.ex. neurala nät)?
Övrigt	Övriga kommentarer.
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	I detta fält anges huruvida modelleringsarbete med beskriven arkitektur har resulterat i några vetenskapliga artiklar.
Status på utvecklad mjukvara	Vilken utvecklingsstatus är det på arkitekturens mjukvara?.
Mjukvaran spridd	Här anges huruvida mjukvaran används av några andra än de ursprungliga tillverkarna.
Används i simuleringar	Används arkitekturen i några simuleringar som är av den komplexitetsgraden till vilken CGF:er kan tänkas behövas?
Används i distribuerade simuleringar	Används arkitekturen i några distribuerade simuleringar när några entiteter styrs av agenter modellerade med arkitekturen. Innebär i dagsläget oftast att arkitekturen har ett gränssnitt som kopplar mot HLA.
VVA-processen	Hur fungerar VV&A processen för modeller utvecklade med arkitekturen?
Modellbyggarens gränssnitt	Vad finns det för stöd för de domänexperter som skall utveckla agenternas beteende och vad finns det för stöd för programmerare som skall koppla mot simuleringsmiljön? Hur svårt är det att skriva koden? En generell observation är att det oftast är själva kunskapsuppbyggnaden och utvecklingen av den konceptuella modellen (se avsnitt 4.3 och 4.4) som är det mest kostsamma delen av en CGF. Verktyg som stöder modellutvecklingen och som tillåter domänexperter att enkelt deltaga i utvecklingen är därför mycket viktigt.
Återanvändbarhet/ utbyggbarhet	Hur mycket av kod och beteende kan återanvändas när en ny agent skall utvecklas? Hur snabbt går det att lägga till nya beteenden och hur mycket behövs läggas till när ett beteende skall göras mer detaljerat?

Integrerbart	Hur fungerar det att integrera agenter eller beteendet modellerade med aktuell arkitektur med andra scenariogenereringsverktyg eller simuleringsmiljöer?
Direkta prestationsmått	
Beräkningskapacitet	Här anges vad arkitekturen kräver för beräkningskapacitet och hur många entiteter som kan hanteras.
Övrigt	
Tillgänglighet	Vad är det för status på arkitekturens tillgänglighet? Är den kommersiell mjukvara, shareware eller ägd av någon myndighet?
Hårdvara/mjukvara/OS	Här anges om det krävs någon särskild mjukvara eller hårdvara.
Huvudsaklig finansiär	Varifrån kommer den huvudsakliga finansieringen?
Omfattning finansiering	Hur omfattande var finansieringen under 2001?
Utvecklare	Var sker den huvudsakliga utvecklingen av arkitekturen?
Användare	Vem/vilka är det som använder arkitekturen?
Användningsområde	Vilken typ av modelleringsprojekt har arkitekturens huvudsakligen använts till?
Tillämpningar/exempel på projekt	Namn på projekt/tillämpningar där arkitekturen används.
Referenser	

Tabell 1. Värderingskritierer för jämförelse av arkitekturer

Downes-Martin (1995) nämner tre centrala frågor när man skall analysera och värdera en CGF-tillämpning:

- Beteendets källa. Hur har utvecklarna samlat in kunskap om hur de aktörer som modelleras agerar i verkligheten. Olika metoder som används är fältstudier och försök, analys av doktrin (observera att doktrin beskriver ett preskriptivt beteende, d.v.s. hur operatörerna bör göra), intervjuer med experter, analyser av loggar (i ett svenskt projekt skulle det kunna vara analys av UTA/UTB loggar), analyser av simuleringar och automatisk maskininlärning.
- Implementationsmetod. Hur har den exekverbara koden konstruerats? Hur fungerar inferens-maskinen d.v.s. hur fungerar själva resonerandet och hur ser själva kunskapsrepresentationen ut? Några exempel som beskrivs i denna rapport är finita tillståndsmaskiner av olika typer, artificiella neurala nätverk och regelbaserade system.
- Teoretiska modeller. Vad finns det för underliggande teoretisk modell av det mänskliga beteendet? I Downes-Martins rapport sägs bara mycket få militära CGF-tillämpningar ha någon underliggande modell av människor som informationshanterare, d.v.s. man har fokuserat på beteendemodeller, inte kognitiva modeller vilket ger problem när systemet skall skalas upp eller anpassas till nya förutsättningar. Under åren som gått sedan Downes-Martins rapport har därför också intresset för kognitiv modellering vuxit.

I DMSO:s Master Plan (DMSO, 1995) nämns under avsnittet Human Behavior Representation följande krav:

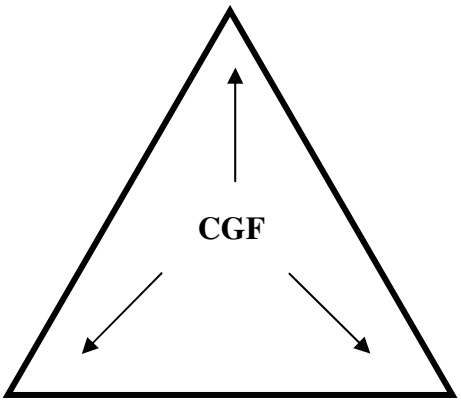
- ”Extend to individuals and to groups”. Det skall gå att modellera både individers och gruppers beteende i arkitekturen. Verksamhet som syftar till att kunna koppla ihop simuleringar med skarpa övningar måste fylla detta krav. Detta eftersom t.ex. en förare av en enskild plattform inte kan slåss mot en aggregerad bataljon utan måste slåss mot de individer som utgör bataljonen, dvs det ställs krav på förmåga till aggregering/deaggregering (Holm & Moradi, 1997).
- ”Generic behaviors”. Det skall finnas generiska modeller av människors förmågor och begränsningar (både för psykologiskt och fysiskt beteende).
- ”On-demand models”. Det skall gå att snabbt att konstruera modeller av individers eller gruppers beteende inför specifika tillämpningar och studier. Detta innebär att det måste finnas många

generella moduler som snabbt går att koppla ihop för att skapa ”ett fullständigt beteende”, d.v.s. att den syntetiska entiteten klarar av att agera i alla situationer den kommer att ställas inför.

2.2. Olika ansatser – olika intressefokus

I utformningen av en CGF-tillämpning finns det ett antal olika ställningstaganden som måste göras vad gäller applikationens utformning. Dessa kan mycket grovt klassificeras i tre grupper vilket visualiseras i triangeln i Figur 1. De ideala CGF-tillämpningen uppfyller givetvis högt ställda krav inom alla delar av triangeln men problemet är att tillfredställande lösningar ännu saknas i alla riktningar. Den ideala CGF tillämpningen innehåller validerade kognitiva modeller av allt mänskligt beteende, är enkel att utveckla och underhålla modeller i, ställer inga alltför stora krav på mjukvaran eller hårdvaran och agenterna kan lösa alla problem på ett smidigt och realistiskt sätt. Dessutom är de syntetiska entiteterna agenterna planerande och kan lära sig nya beteenden (om det är önskvärt). Innan det är möjligt att uppfylla detta återstår dock många års forskning.

Intressefokus AI
Problemlösningsförmåga.
Kunskapsrepresentation.
De syntetiska entiteterna skall kunna lösa ”alla” problem.



Intressefokus Kognitiv modellering
Modellernas skall vara kognitivt korrekta.
Beteendet och beslutsfattandet överensstämmer med experimentella data och gängse teorier.

Intressefokus MoS
Effektiv simuleringsprocess.
”Ekonomisk” modellutveckling.

Figur 1. Många hänsynstaganden och avvägningar styr utformningen av en CGF tillämpning.

De modelleringsarkitekturer och problemlösningsansatser som beskrivas i appendix A, kan sägas ha sitt intressefokus i något av triangelns hörn, även om de givetvis vill täcka så mycket som möjligt på ett tillfredställande sätt, så nedanstående klassificering är en grov förenkling. De kognitiva modelleringsarkitekturerna har intressefokus kognitiv modellering, scenariogenereringsverktygen och ”state-flow” verktygen har intressefokus MoS och de ansatser för problemlösning och kunskapsrepresentation som beskrivs har intressefokus AI.

Klassificering	**Namn**	**Intressefokus**
CGF-verktyg utan kognitiva modelleringsanspråk	VR Forces Agent Factory MicroSaint	MoS
Kognitiva Modelleringsarkitekturer	ACT-R och ACT-PM Soar, EPIC EPIC-Soar iGen/Cognet MIDAS	Kognitiv Modellering
Scenariogenereringsverktyg	Flames Stage STRIVE Tacsi ModSAF (inkl OneSAF och JSAF)	MoS
AI-tekniker för kunskapsrepresentation (ett mycket begränsat urval)	ANN (Artificiella neurala nät) Case based Reasoning Genetiska Algoritmer Fuzzy Systems Finita tillståndsmaskiner (olika typer) Regelbaserade system (Production systems)	AI

Tabell 2. Grov kategorisering av beskrivna modelleringsansatser.

3. Behovet av kognitiv modellering

Detta avsnitt är avsett att förklara varför framtidens CGF utveckling kommer att kräva ett större inslag av kognitiv modellering än vad som finns i flertalet av dagens CGF-tillämpningar.

För att CGF:er skall få ett i alla avseenden korrekt beteende kommer det att behövas modeller av bl.a. följande psykologiska ”fenomen” och faktorer:

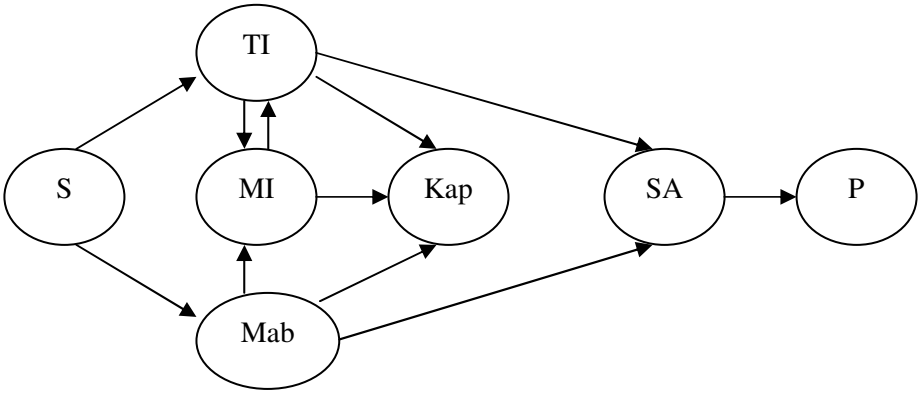
- Perception och uppmärksamhet
- Motoriskt beteende
- Minne
- Situationsmedvetande
- Mental arbetsbelastning
- Motivation
- Sociala faktorer
- Förmåga att lösa flera uppgifter samtidigt (multi-tasking)
- Beslutsfattande
- Inlärning
- Kommunikation mellan personer
- Känslor som t.ex. rädsla

För vissa tillämpningar, som t.ex. enklare träningssyften behöver inte alla dessa faktorer finnas modellerade. I litet mer krävande tillämpningar, som t.ex. om CGF:er skall användas i en materialanskaffningsprocess för hjälmsikten så kommer det att bli viktigare.

En annan anledning till att ha experimentellt grundade teorier och modeller för t.ex. beslutsfattande i en CGF-tillämpning är att det underlättar när man vill skala upp ett CGF-system eller anpassa det till en annan domän. Om det inte finns någon underliggande teori så är alla regler o.s.v. utformade för att ge ett utåt sett trovärdigt beteende i den aktuella simuleringen, men de tenderar att vara ”datahack” för att få simulationen att snurra och kommer då inte gå att skala upp eller anpassas till de nya förutsättningarna.

Inom den beteendevetenskapliga forskningen finns modeller för ovan uppräknade fenomen, men ibland är de dock för vaga eller generella för att direkt kunna infogas i en CGF-tillämpning. Inom detta område krävs mer forskning och utvecklingsarbete vilket också poängterades vid ”10th Conference on Computer Generated Forces and Behavioral Representation”.

Exempel på avancerade beteendevetenskapliga modeller som är mer rättframma att infoga i en CGF-tillämpning beskrivs i t.ex. Svensson mfl (1992 och 1997). Ett exempel på dylika modeller visas i figur 2. De experimentella underlaget för denna modell av hur informationskomplexitet, situationsmedvetande, mental arbetsbelastning och operativ prestation korrelerar, är insamlat under verkliga träningsuppdrag i luften. Även dessa modeller kräver dock viss anpassning innan de går att implementera direkt.



S = Uppdragets svårighetsgrad
TI = Informationskomplexitet på den taktiska indikatorn, TI
MI = Informationskomplexitet på målindikatorn, MI
Mab = Mental arbetsbelastning
Kap = Mental reserv kapacitet
SA= Situationsmedvetande, ”koll på läget”
P= Prestation

Figur 2. Modell av kausala samband mellan en mängd beteendevetenskapliga konstrukter och objektiva. Observera att alla länkar i ovanstående LISREL diagram är viktade vilket inte framgår av figuren.

4. Prototyputveckling

Inom verksamheten under 2001 modellerades en enkel modell av en flygförare som utför ett attackuppdrag i två arkitekturer, Soar och Agent Factory. I detta avsnitt redovisas resultatet av arbetet och erfarenheter från att bygga datorgenererade piloter med hjälp av experimentellt grundade beslutsmodeller.

Under det senaste året har ytterligare tre prototyper tagits fram. De fem prototyperna är beskrivna i appendix B. Utveckling av gränssnitt till Soar beskrivs i avsnitt 4.2 och ett förslag till hur man bygger ett komponentbaserat bibliotek med ”plug’n’play” funktioner finns i avsnitt 4.5.

Skälen till att flygdomänen har valts för att bygga datorgenererade piloter är följande:

- Projektdeltagarna har mest domänkännedom inom detta område.
- Projektet har tillgång till domänexperter, d.v.s. flygförare på FLSC och F17.
- FLSC är den simulatoranläggning i Sverige som i dagsläget har störst behov av mer avancerade CGF:er.
- Genom att en av projektmedlemmarna även deltar i FOI projektet ”Mental belastning och prestation” finns tillgång på experimentella data och beslutsmodeller.

Modelleringsarkitekturerna Soar och Agent Factory har valts eftersom:

- TAC-Air-Soar har visat att Soar är en framkomlig väg. Soar är dessutom mest spridda modelleringsarkitekturen.
- Agent Factory är ett svenskt utvecklingsprojekt finansierat av FMV. Agent Factory representerar ett mycket modernt tänkande vad gäller bl.a. stöd för modelleraren och stöd för distribuerad simulering.

4.1. Användarjämförelse Soar - Agent Factory

De två arkitekturer som projektet studerat närmare har likartade mekanismer och egenskaper. I detta avsnitt har vi utifrån modellutvecklarens perspektiv gjort en jämförelse mellan dessa två arkitekturer som belyser skillnader mellan samt styrkor och svagheter med dessa arkitekturer. Som egenskaper räknas här ej prestandafaktorer som beräkningshastighet eller minnesåtgång.

4.1.1. Agenten i världen d.v.s. ”Situatedness”

Denna term används för att beskriva möjligheterna att modellera att agenten befinner sig i en värld, t.ex. att en pilot styr genom att hantera sin styrspak och får sin information genom att titta ut och titta på displayer.

Agent Factory
Eftersom Agent Factory har stöd för koppling till HLA-arkitekturen sker all kommunikation mellan agenter utvecklad i denna arkitektur och omgivningen via RTI. All information som skall kommuniceras med omvärlden beskrivs först i en konceptuell modell. Det finns inget behov av att översätta denna information till en struktur som består av arbetsminneselement som i Soar då uppdateringen av arbetsminnet sker automatiskt. För att hantera informationsfiltrering kan man lägga till sensor- och/eller motorsystem komponenter (moduler) till RTI. En modul kan användas för att hantera filtrering av visuell information medan en annan hanterar hörselfiltrering eller motoriken.

Modulerna kan dessutom kommunicera och påverka varandra för att skapa mer komplexa filtreringssystem. Denna komponentbaserade arkitektur erbjuder stora återanvändningsmöjligheter.

Soar
För att Soar agenter skall kunna kommunicera med omvärlden måste de förses med en extern kommunikationsmodul. Modulen måste kunna ta emot objekt- och attributinformation från omvärlden och översätta dessa till arbetsminnesstrukturer. Översättningsprocessen kan bli mycket komplex för stora system. Statusändringar och dylikt hos agenten skall också översättas och kommuniceras med omgivningen. För stora mängder av in- och/eller utdata, kan detta bli en stor utmaning även om Soar-arkitekturen erbjuder viss funktionalitet för att skapa sådana moduler i C eller Tcl kod. Det finns också ett antal externa paket som förenklar kommunikation och strukturering. I vissa fall, måste även beteendet hos agenten utökas med regler som övervakar utdataprocessen.

4.1.2. Kunskapsrepresentation

Här har syftet varit att undersöka hur kunskap representeras i de olika arkitekturen, t.ex. hur arbetsminnet är representerat.

Agent Factory
Arbetsminnet i Agent Factory är representerat av objekt med tillhörande attribut. Alla objekt som agenten skall kunna resonera om måste beskrivas på förhand genom att skapa klasser och attribut i en konceptuell modell. Arbetsminnet är därmed statiskt och nya klasser eller elementtyper kan varken läggas till eller tas bort. Om det dyker upp objekt som inte är definierade kan agenten sluta fungera. Agenten kan nämligen inte resonera om okända objekt eller attribut. Detta kan vara olämpligt då den interna resoneringsprocessen bör kunna skapa temporära abstraktioner som är endast meningsfulla i den aktuella situationen. Under en luftstridssituation kan agenten t.ex. behöva skapa abstraktioner som ”avståndet är tillräckligt”, ”lite lågt” eller ”vinkel inför attack är okej”. Agenten skall då resonera om abstraktionerna och inte aktuella siffror. För att dessa abstraktioner skall kunna användas behöver de definieras i den konceptuella modellen och därmed kommer de att existera under agentens hela livslängd. För komplexa beteenden kan många abstraktioner behövas och det skulle kunna leda till stora svårigheter för hantering och uppdatering av konceptuella modellen. Dessutom kan agenterna inte resonera om sin aktuella eller tidigare status i Agent Factory. För att åstadkomma detta måste ytterligare objekt skapas i konceptuella modellen.

Soar
Arbetsminnet i Soar definieras av symboler, som är kopplas till varandra och därigenom bildar en trädstruktur av objekt och attribut. Symbolträdet är mycket dynamiskt och behöver inte definieras på förhand. Agenten kan lägga till, ta bort, eller ändra element i minnet och koppla ihop symbolerna på ett godtyckligt sätt. Dessutom är det pågående målet eller operatorn en del av arbetsminnet, som möjliggör det för agenten att resonera om sin aktuella eller tidigare status. Symboler modifieras antingen genom internresonering av agenten eller via en kommunikationsmodul. Agenten har oftast ett delträd av symboler som tar emot perceptioner från omgivningen. Dessa symboler vanligtvis struktureras och modifieras av indatamodulen.

4.1.3. Beteenderepresentation

Detta avsnitt betraktar flödeskontrollmekanismer som producerar beteende.

Agent Factory
I Agent Factory består långtidsminnet eller resoneringsdelen av separata villkor- och slutsatskomponenter. Komponenterna kan användas i generella regler som anropas så fort arbetsminneselement har ändrats och tillfredställer villkorsdelen. Det så kallade mål-plan-fas trädet (goal-plan-phase tree) sätter upp ett målorienterat beteende, och villkorskomponenter hjälper

inferensmaskinen till att gå genom trädet. Mål-plan-fas trädet är statiskt vilket gör att mål, planer och faser inte kan återanvändas i andra delar av trädet, utan måste definieras om för alla förekomster.

Soar
Långtidsminnet i Soar består av produktionsregler som är kategoriserade som vanliga (I-supported) regler och operatorer (O-supported). Eftersom Soar använder en automatisk delmålsmekanism, som dynamiskt väljer ett mål, kan det vara svårt att förstå eller förutsäga beteendet. Mekanismen erbjuder dock ett mycket dynamiskt beteende, där många operatorer kan utvärderas i ”look-ahead” sökningar. Därmed har man i Soar möjligheten att slumpmässigt välja ett mål när operatorer har samma värde. Det automatiska framtagandet av delmål gör det möjligt att återanvända operatorer och delmål. Detta möjliggör även användning av bibliotek av komplexa operatorer som kan anropas i olika situationer (t.ex. en komplex manöver som ”zig-zag”, som består av flera delmålsoperatorer kan användas i olika situationer när man t.ex. undviker fientliga robotar). Soar-agenter har också förmågan att lära sig nya regler vilken kan vara användbar i framtida utveckling av komplexa system.

4.1.4. Gränssnitt mot modellerare/utvecklare

Här har arkitekturerna granskats i sin roll som verktyg för utveckling av beteende och vad modellutvecklaren har för användargränssnitt till stöd för agentutvecklingen.

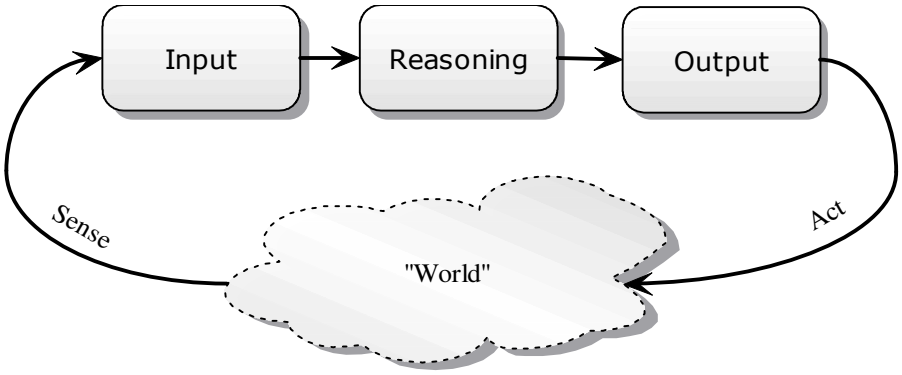
Agent Factory
Agent Factory har ett grafiskt användargränssnitt. Villkor- och slutsatskomponenterna ges ett namn och har ett valfritt dokumentationsfält vilket förenklar för icke-programmerare att använda verktyget. Det finns också en grafisk representation av mål-plan-fas trädet. Eftersom trädet är statiskt är det enkelt att följa vägen genom det och förutse det resulterade beteendet.

Soar
Soar har en egen syntax för att skriva regler som kanske inte är så enkelt att förstå för en icke-programmerare. Det finns inte heller någon grafisk representation av beteende eller arbetsminne. Eftersom båda representationerna i Soar är mycket dynamiska kan det visa sig vara svårt även för en Soar-programmerare att förutse beteendet.

4.2. Utveckling av gränssnitt till Soar

4.2.1. Bakgrund

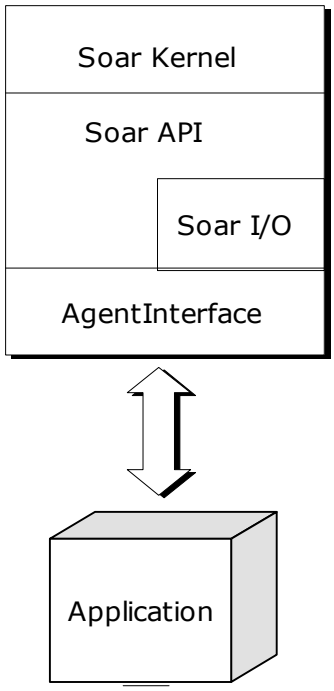
Soar är en AI-arkitektur skriven i programspråket C. Arkitekturen använder sig av flödet ”input-reasoning-output”. I ”input”-fasen hämtas information om omvärlden från t.ex. en simulator. Därefter resonerar Soar agenten om den hämtade informationen samt information som lagrats i det egna minnet. Beroende på vad den resonerande delen kommit fram till kan agenten välja att skicka information tillbaka till simulatorn i form av kommandon som t.ex. förflyttningar. Detta flöde upprepar sig i cykler. Detta tillvägagångssätt används i flera AI- och kognitiva arkitekturer.



Figur 3.Typiskt flöde i kognitiva- och AI- arkitekturer.

4.2.2. Utveckling

Genom input- och output faserna kan en Soar agent interagera med en simulator eller annat program. Det blir då programmets uppgift att skicka och hämta information till respektive från agenten. Soar tillhandahåller ett gränssnitt skrivet i C för att kunna utföra detta. Gränssnittet består av ett antal lågnivå funktioner som kräver mycket god kännedom om Soar's funktionalitet för att kunna användas. För interaktion mellan Soar och t.ex. simulator är man ofta tvungen att skriva en del egna rutiner utöver dessa lågnivå funktioner. Dessa rutiner upprepar sig oberoende av tillämpning. För att slippa skriva om dessa rutiner och för abstrahera lågnivå funktionaliteten har det skapats nya gränssnitt som förenklar arbetet med interaktion. Det mest uppmärksammade gränssnittet heter SGIO och har en objektsorienterad struktur, skriven i C++. SGIO har bl.a använts i projektet för att koppla ihop Soar med spelet Quake 2. Utifrån erfarenheterna med SGIO och Soar's lågnivå gränssnitt skapade projektet ett nytt gränssnitt som för närvarande lever under namnet AgentInterface (förkortat AI).



Figur 4. Agentinterface.

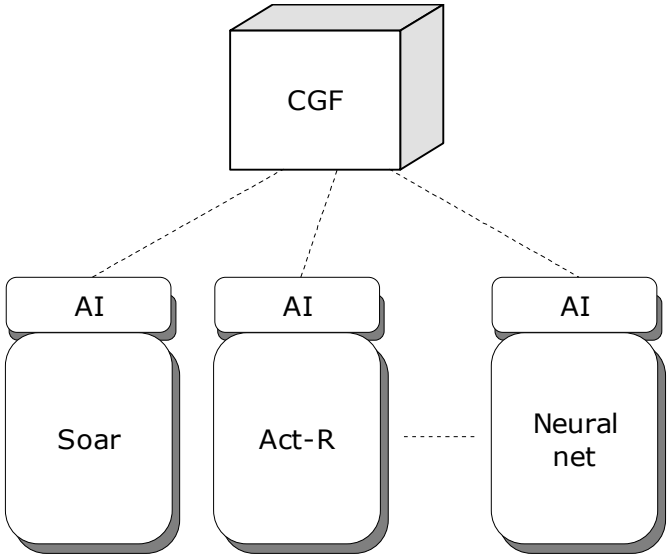
AI är skrivet i C++ och använder en objektsorienterad struktur där agenten är satt i fokus. Agenten är en klass innehållande tillståndsvariabler och funktioner för att känna av och påverka omvärlden. Denna fokusering på agenten gör att det blir lätt att ärva funktionalitet, skapa och driva flera agenter samtidigt. AI utnyttjar Soar's lågnivå gränssnitt och har en liknande struktur som SGIO men skiljer sig på några punkter:

- SGIO stödjer bara Windows, AI stödjer Windows och många Unixmiljöer.
- I AI kan simuleringen antingen drivas av agenten eller systemet, i SGIO bara systemet.
- I AI kan man dirigera om utskrifter från Soar till fönster, spelkonsoler, filer mm. Går ej i SGIO.
- I AI har man kontroll över input- och output-faserna. Dessa är gömda i SGIO.
- I AI finns funktionalitet för att konvertera XML-filer till minneselement, Finns ej i SGIO.
- I AI kan man skicka alla Soar kommandon via konsol eller liknande. Går ej i SGIO
- SGIO har en grafisk editor kallad TSI för bl.a. felsökning, denna funktionalitet går dock bara att utnyttja via Sockets (Soar och simuleringsprogrammet måste köras på skilda datorer). AI har för närvarande ingen grafisk miljö.

Agentinterface har bl. a används i prototyperna Flames-Pilot och Acm-Agent. (Se appendix B)

4.2.3. Framtid

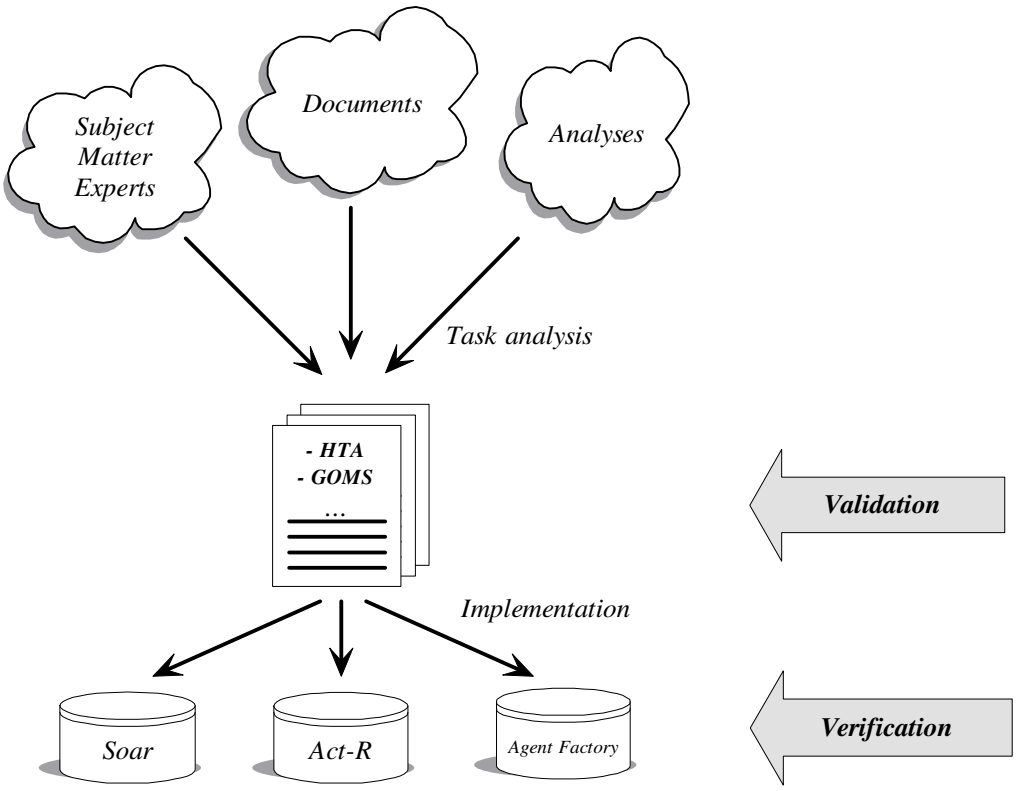
Det generella namnet AgentInterface härrör från en ide att skapa ett uniformt gränssnitt för flera AI- och kognitiva arkitekturer. Detta gör att man kan byta ut arkitekturerna utan att göra några ändringar i programkod, då koden bara förutsätter ett specifikt gränssnitt. Generaliteten kommer från det faktum att de flesta arkitekturer försöker att modellera intelligens eller en människas sätt att arbeta (se Newell, 1990). De utnyttjar således varianter av ett ”input-reasoning-output” flöde. Arkitekturerna har också mekanismer för att lagra minneselement i ett s.k. working memory. Vid avvikande arkitekturer går detta att simulera, t.ex. för neurala nät. AgentInterface stödjer detta synsätt på beteende mekanismer och bör därför kunna utgöra ett generellt gränssnitt för en mängd arkitekturer. Vid utnyttjandet av ett sådant gränssnitt förenklas arbetet avsevärt för en utvecklare av datorgenererade styrkor och bör därför finnas tillhands i projektets bibliotheksstruktur.



Figur 5. Agentinterface som generellt gränssnitt till flera arkitekturer.

4.3. Uppgiftsanalys

Uppgiftsanalysen och ”knowledge engineering/acquisition” processen är helt central för utformning av användbara CGF:er. För att skapa ett realistiskt pilotbeteende, måste den modellerade operatörens uppdrag eller uppgift analyseras och beskrivas. Experter inom den aktuella domänen besitter kunskapen, men det finns olika metoder för att utvinna denna information. Diskussioner, intervjuer och noggranna fältstudier är några av metoderna för anskaffning av information.



Figur 6 .Uppgiftsanalys i CGF-utvecklar processen.

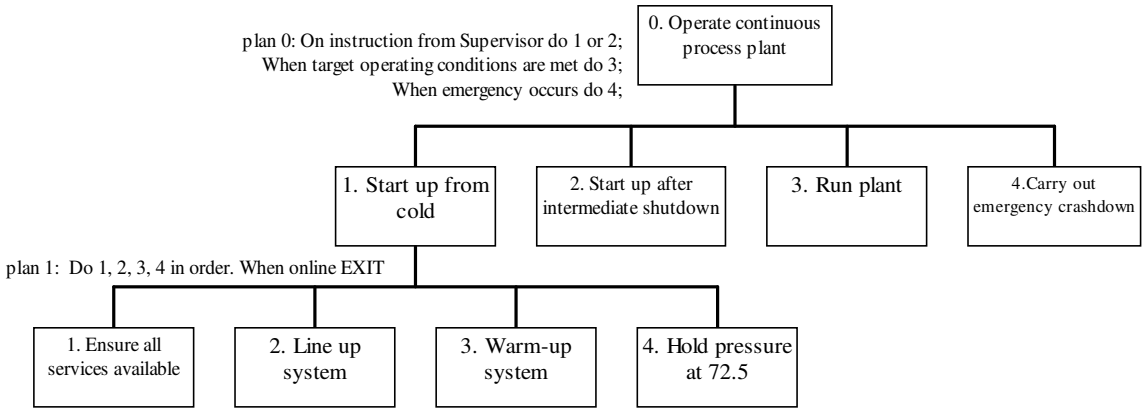
För den agent som implementerades inom projektet användes uppgiftsanalysmetoden HTA (Hierarchical Task Analysis) och en analys av ett attackuppdrag gjordes tillsammans med flygförare på F17 med stöd av uppdragsloggar på UTB band. UTB och UTA bandspelare erbjuder en mycket sofistikerad inspelningsutrustning för kontinuerligt loggning av en mängd intressanta parametrar under verkliga flyguppdrag. Efter landning kan loggarna spelas upp för analys och indikatorer och skärmar kan analyseras för att i diskussion med flygförare diskutera hur de tänkte eller varför de gjorde som de gjorde.

Andra metoder för uppdragsanalys är GOMS (avsnitt 4.3.2) och CTA, Cognitive Task Analysis.

4.3.1. HTA (Hierarchical Task Analysis)

Hierarchical Task Analysis (Kirwan & Ainsworth, 1992) är en uppgiftsanalysmetod som används för att skapa beskrivningar av uppdrag och mänskliga verksamheter i termer av mål, operationer och planer. Mål står för systemets önskade läge, t.ex. förstörelse av en bro eller ett luftrum utan fientliga flygplan. En operation är en beteendeenhet som exekveras för att uppnå ett mål. Denna beteendeenhet

kan beskrivas i olika komplexitets- och varaktighetsnivåer. I en kontext kan en operator vara att patrullera på ett flygfält, medan en annan operator kan vara att röra en styrspak. Båda operationerna kan vara delar av samma mål, men vara beskrivna på olika detaljnivåer. En plan anger det villkor som bestämmer när varje operation skall utföras. Planer inkluderar ofta en rad handlingar som är kopplade till tid eller processvillkor. Ibland behöver högnivåoperationer delas i ett antal deloperationer och vara delar av ett delmål. Uppdelningsprocessen kan fortsätta tills önskad detaljnivå har uppnåtts. Många kognitiva teorier påstår att alla organismer ständigt försöker att uppnå ett antal mål som i sin tur är delar av högre mål och det var därför HTA tekniken ansågs vara en lämplig beskrivningsmetod.



Figur 7 .Exempel på HTA för att sköta en reaktor

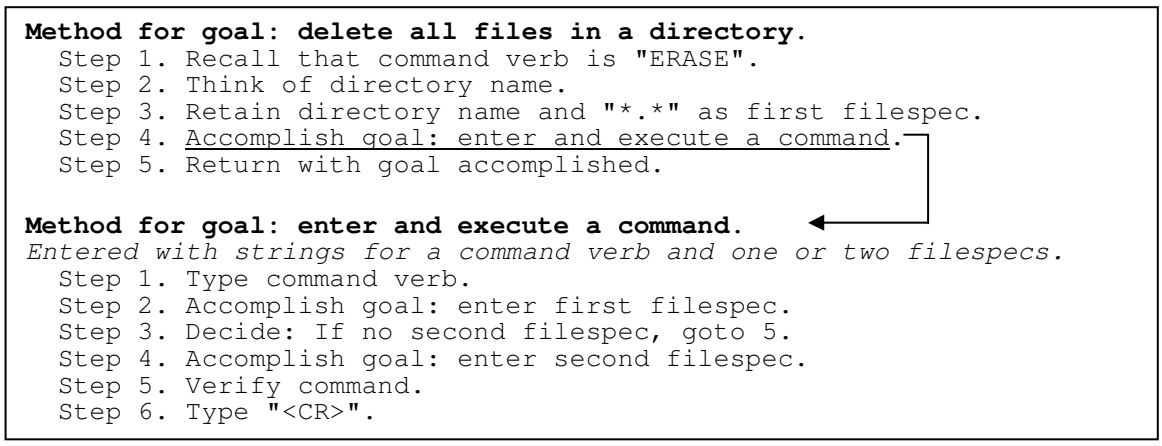
Genom att använda en effektiv dokumenteringsmetod, som både experter och modellutvecklare lätt kan begripa, förbättras både datainsamling och valideringsprocessen. Dokumentering skapar också ett lager mellan experter och implementation, vilket möjliggör att skapa olika modeller och implementationer från samma datamängd. HTA metoden som användes här erbjöd ett tillräckligt bra verktyg för dokumentering som både experter och modellutvecklare kunde förstå.

4.3.2. GOMS

En GOMS-modell är en beskrivning av den kunskap en användare måste ha för att kunna utföra uppgifter genom ett system (se Kieras, 1994). GOMS står för Goals, Operators Methods and Selection Rules. GOMS är alltså en beskrivning av de metoder (methods) som behövs för att uppfylla ett specificerat mål (goal). Metoderna är uppbyggda av en serie operatorer (operators) och delmål så att dessa bildar en hierarkisk struktur. Om det finns flera metoder för att uppfylla ett mål finns det också regler som väljer ut en av metoderna (selection rules).

I likhet med HTA kräver GOMS att analyseraren identifierar mål, delmål, operatorer och regler. Lika viktigt är att man identifierar sådant som inte skall tas utifrån de krav som ställts på modellen. GOMS finns i flera olika dialekter. Den kanske vanligaste dialekten för kognitiv modellering är ”Natural” GOMS Language, NGOMSL. NGOMSL har en direkt relation till regelbaserade system, kräver inte någon kunskap om det underliggande regel-systemet. Till skillnad från t.ex. HTA så har GOMS en formell notation vilken kan direkt exekveras av en användare eller ett GOMS-interpreterande program. Ibland beskrivs GOMS som ”sinnets programmeringsspråk” (”The programming language of the mind”). Detta medför bl.a. att man i NGOMSL inte kan skriva komplicerade villkorssatser eller komplexa algoritmer eftersom människan inte förmodas klara av sådan behandling i ett enda kognitivt steg. NGOMSL bygger också på teorier om människans minnesrymder, arbetsminne (working memory) och långtidsminne (long term memory), genom att stödja manipulation av dessa i modellbeskrivningen.

Eftersom GOMS har en formell notation, kan det vara svårt att läsa av ”koden” för att försöka förstå syftet med metoderna. Därför bör en GOMS beskrivning åtföljas av någon annan, icke exekverbar, dokumentationsform.



Figur 8. Exempel på NGOMSL för att radera alla filer i en mapp

4.4. VV&A

För att simuleringar där CGF:er deltager skall vara användbara måste de simulerade styrkornas beteende efterlikna det beteende som mänskliga operatörer uppvisar och det kommer att krävas en omfattande VV&A process innan framtidens svenska CGF:er kan ackrediteras.

En intressant fråga i VV&A sammanhang är huruvida man kan uttala sig om korrekt uppförande enligt doktrin om man använder sig av modeller som ibland tar felaktiga beslut p.g.a. för hög mental arbetsbelastning. En preskriptiv modell som beskriver hur operatören borde göra är mycket lättare att utvärdera än en deskriptiv som beskriver hur det faktiskt går till.

Det första steget i ett VV&A arbete är att ta fram en konceptuell modell för den tänkta CGF:en.
Den konceptuella modellen beskriver tre punkter som är väsentliga för utveckling och implementering av simuleringsmodellen:

- Kontexten i vilken CGF:erna byggs, t.ex. flygstridsdomänen.
- Information som beskriver vad som påverkar CGF:ens beteende som t.ex. status, händelser, prestanda.
- Information om den totala simuleringen, t.ex. hur simuleringsenheter interagerar och kopplas ihop.

4.5. Ett komponentbaserat bibliotek av datorgenererade styrkor.

4.5.1. Introduktion

En aktivitet som bedrivs under projektet ”Datorgenererade styrkor” är att undersöka möjligheten att skapa ett försvarsgemensamt bibliotek för datorgenererade styrkor. Tanken är att biblioteket ska innehålla ett antal ”plug’n’play” *komponenter* som en användare enkelt ska kunna integrera med befintliga simuleringsramverk, spel, simulatorer, mm. Biblioteket skall innehålla CGF-modeller för olika domäner, i olika aggregationstillstånd och med varierande detaljnivåer. En användare skall således kunna välja den typ av CGF-modell som behövs och stoppa in i sin simuleringsmodell.

Det förslag på bibliotek som ges här, inkluderar en extern och en intern struktur. Den externa strukturen skall möjliggöra ”plug’n’play” kompatibilitet, dvs göra det enkelt att använda CGF-modeller i olika applikationer. Om den externa strukturen beskriver hur man ska använda en CGF-modell, beskriver den interna strukturen hur man kan skapar CGF-modeller. Den interna strukturen är ett hjälpmedel för att på ett standardiserat sätt skapa modeller men är inte ett krav, dvs man kan skapa modeller som inte bygger på den interna strukturen, bara de följer de regler som följer av den externa strukturen.

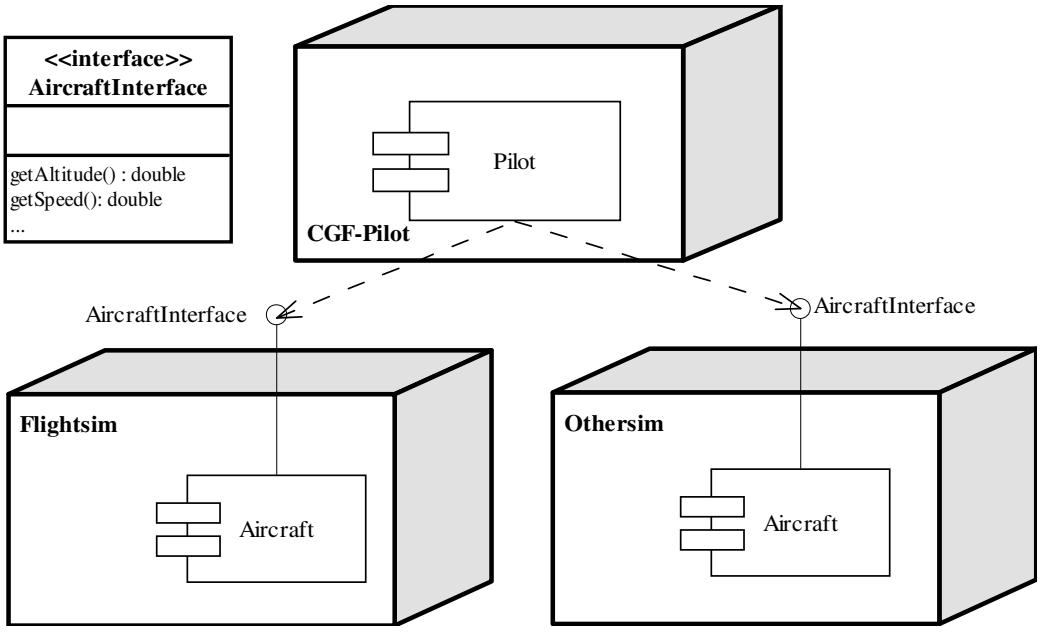
4.5.2. Komponenter

För att åstadkomma en komponentbaserade struktur krävs det att man specificerar standardiserade gränssnitt så att komponenter kan interagera med varandra. Om en komponent ska utnyttjas av andra komponenter så måste komponenten *erbjuda* ett standardiserat gränssnitt. Alla andra komponenter som vill utnyttja komponenten vet då vilket eller vilka gränssnitt som de kan förvänta. En komponent som utnyttjar andra komponenter *kräver* vissa gränssnitt av dessa för god funktionalitet. Eftersom komponenterna enbart interagerar via standardiserade gränssnitt behöver de inte veta exakt hur de andra komponenterna fungerar. Detta är nyckeln i komponentbaserade system. Komponenterna kan således bytas ut mot andra komponenter så länge de nya komponenterna erbjuder samma gränssnitt. Komponenter som erbjuder samma gränssnitt grupperas som en *komponenttyp.*

4.5.3. Extern struktur

Den externa strukturen som beskriver en komponenttyp i CGF-biblioteket representerar en komplett entitet som antingen beskriver enskilda spelare (pilot, stridsvagnsförare, soldat, etc) eller grupper (stridspar, pluton, etc).

För att kunna utnyttja komponenttänkandet för datorgenererade styrkor måste man således specificera ett antal standardiserade gränssnitt. Den lösning som valts för detta bibliotek är att låta CGF modellen se applikationen (simulator, spel, mm) som en komponent från vilken den kräver vissa gränssnitt. Applikationen ser således CGF-modellen som en komponent som utnyttjar applikationens tjänster. Det är då upp till applikationen att implementera och erbjuda dessa gränssnitt. Då applikationen skapat dessa gränssnitt, kan den sedan byta mellan olika implementationer av CGF-modeller utan att ändra i programkoden. Detta medför också att då man väl implementerat en CGF-modell kommer denna att kunna återanvändas av andra applikationer som erbjuder de gränssnitt som modellen kräver. På detta sätt kan modellerna återanvändas och bytas ut av applikationerna.



Figur 9. Exempel på en komponent i CGF-biblioteket. CGF-Pilot kräver vissa funktioner från applikationerna som vill använda CGF-Pilot. Dessa funktioner specificeras i gränssnittet "AircraftInterface".De applikationer som erbjuder dessa funktioner kan helt enkelt" plugga-in" CGF-pilot utan att behöva ändra någon kod. CGF-pilot blir således en komponent som kan återanvändas och delas mellan applikationer.

4.5.4. Komponenter i CGF-biblioteket

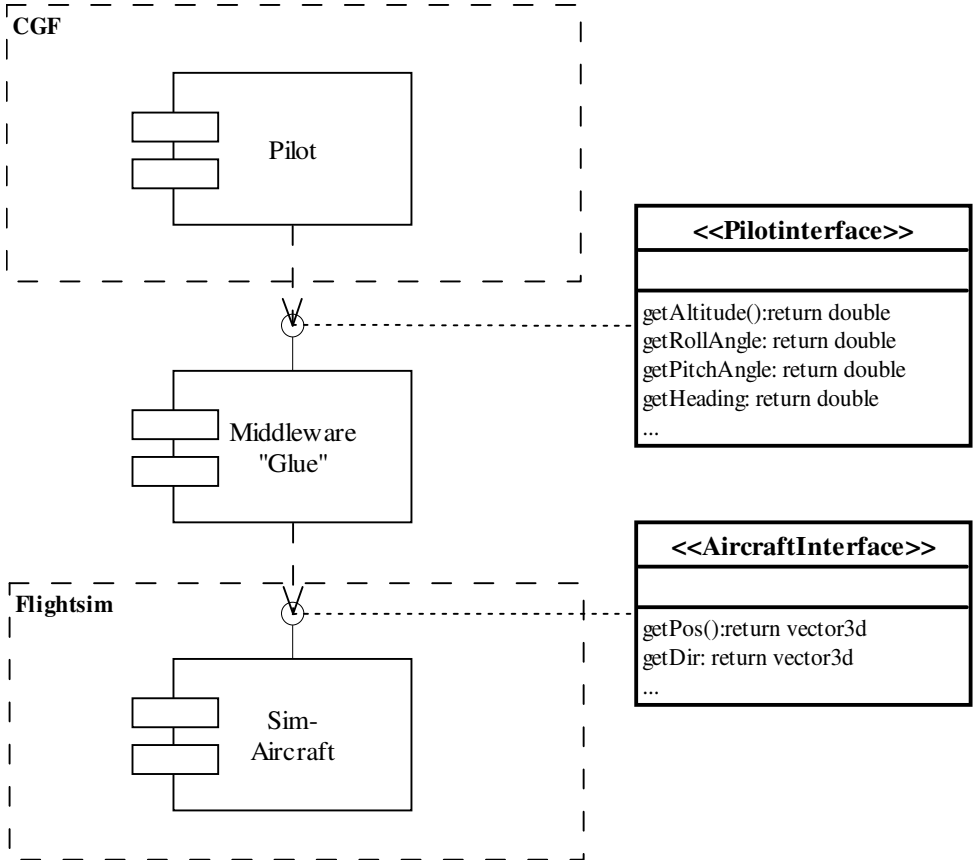
För en utvecklare av en CGF-modell innebär detta att han vet vilka gränssnitt han kan förutsätta från applikationen. Han kan då välja precis vilka funktioner ur dessa gränssnitt som han vill utnyttja. Enklare modeller kan använda ett fåtal funktioner medan mer komplexa modeller kan utnyttja fler funktioner. Hur funktionerna sedan används är helt upp till utvecklaren. Modellutvecklaren kan då välja fritt om han vill välja neurala nät, regelbaserade system, algoritmer eller någon annan arkitektur för att styra beteendet.

För en applikationsutvecklare som vill använda CGF-modellen innebär detta att han måste erbjuda ett visst gränssnitt. Eftersom gränssnittet är specificerat behöver utvecklaren bara implementera gränssnittet. Gränssnittet bör således vara utformat så att implementationen blir enkel och förhoppningsvis bara genererar ett fåtal rader kod för varje funktion. I de flesta fall finns redan funktionaliteten i applikationen, vilket gör att gränssnittet fungerar som en "wrapper". Gränssnittsfunktionerna kommer att vara av sådan typ att de ska kunna realiseras av de flesta applikationer. I de fall där en implementation förefaller vara svår eller rent av omöjlig kan inte applikationen erbjuda gränssnittet fullt ut. Vid sådana fall får applikationsutvecklaren specificera vilka funktioner som stöds. Sådana applikationer kan således bara utnyttja CGF modeller som inte använder de saknade funktionerna.

Alternativ Metod (dubbla gränssnitt)

För att uppmuntra till användandet av biblioteket, är tanken att arbete i form av gränssnittsanpassning av applikationer blir minimal, samt att CGF-modellerna lätt kan hämta de data de behöver. Ett sätt att möjliggöra detta är att skapa ett gränssnitt för applikationen, ett annat gränssnitt för CGF-modellen och ha ett *middleware* som knyter ihop de båda. Detta middleware tillhandahålls av biblioteket.

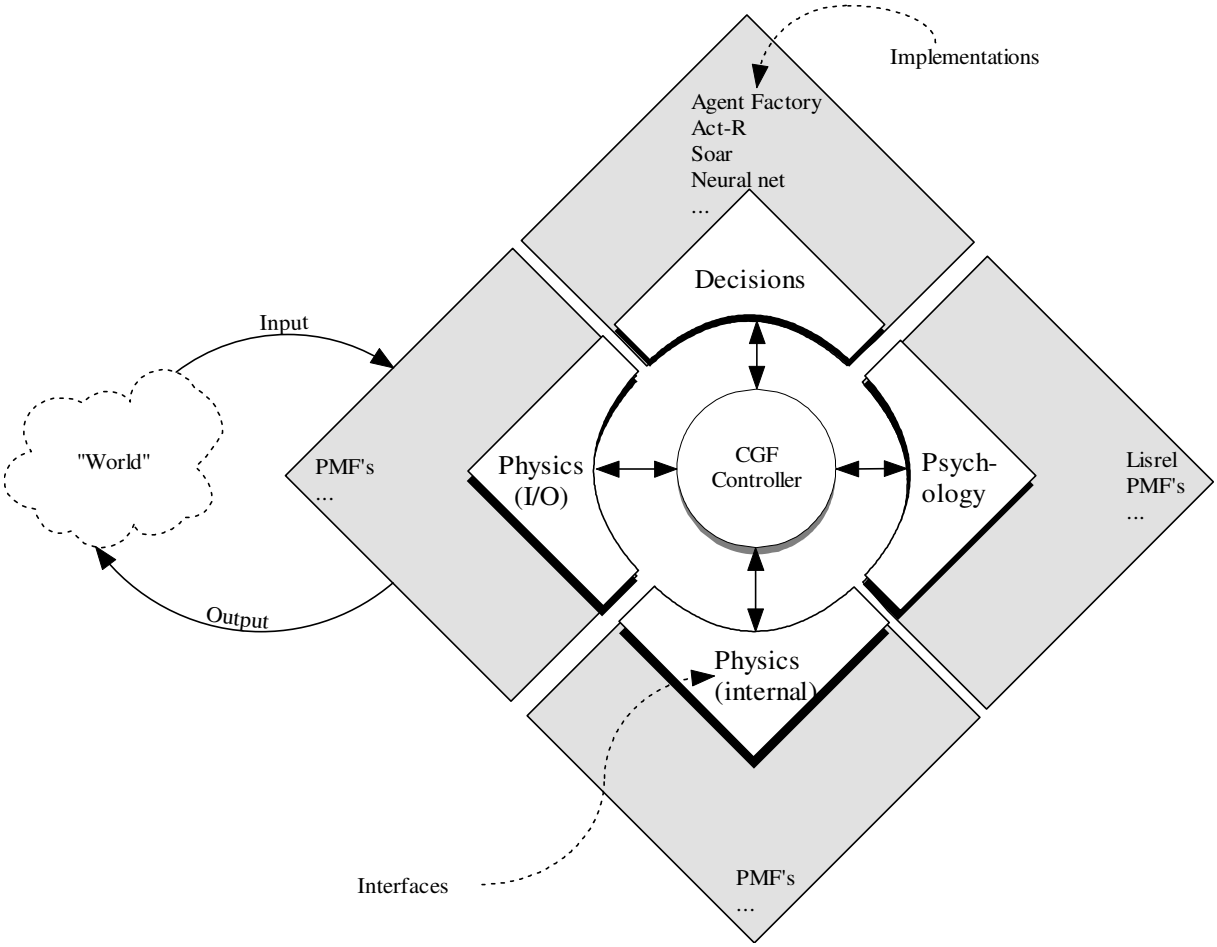
Resultatet blir att applikationsutvecklaren inte behöver erbjuda funktionalitet för varje funktion som en CGF-modell kan tänkas sig att använda, det räcker med att utvecklaren erbjuder ett gränssnitt som CGF-funktionerna kan härledas ifrån. Samtidigt behöver inte CGF-utvecklaren ägna sig åt att konvertera och formatera data från applikationen till de former som erfordras för CGF-modellen. Biblioteket skall också tillhandahålla hjälpfunktioner för att ytterligare förenkla integrationsarbetet.



Figur 10. Förslag på modell för att koppla ihop en applikation med en CGF-modell. Applikationen tillhandahåller funktioner för att hämta vektordata från ett flygplan. Dessa vektorer finns troligtvis redan i applikationen. En pilot resonerar sällan om vektor-data utan om enklare former som skalärer. För att undvika att tvinga flygplanssimulatorn tillhandahålla dessa enklare datafunktioner, finns en översättare som konverterar vektor-data till skalär-data som lättare kan användas av CGF-modellen. Denna översättare tillhandahålls av biblioteket.

4.5.5. Intern struktur

Den interna strukturen för CGF-utveckling är till hjälp för en utvecklare. Strukturen omfattar ett standardiserat sätt på hur olika beståndsdelar kan sättas ihop till en CGF-entitet. Beståndsdelarna kan t.ex. representera fysiska förlopp, psykologi, resonerande vilka interagerar med varandra för att simulera ett beteende. Vilka beståndsdelar som man vill använda är helt valfritt, i vissa fall räcker det med en resonerande del som inte har några psykologiska eller fysiska begränsningar. Genom en sådan struktur kan man byta ut och återanvända sina modeller för psykologiska, fysiska eller resonerande fenomen.



Figur 11. Exempel på en möjlig intern struktur för att utveckla CGF modeller

Resonerande del

Den resonerande delen är den del som fattar beslut. Besluten är ofta representerade med regler (production rule systems), men kan också utgöras av t.ex. neurala nät eller annan kunskapsrepresentation. De flesta resonerande system använder flödet ”input-reasoning-output”, dvs först hämtar systemet information från omvärlden, därefter resonerar systemet om denna information och tar beslut, till sist påverkar systemet omvärlden genom bestämda gärningar.

I de prototyper som skapats under projektet har arkitekturen Soar använts för den resonerande delen. Till Soar har projektet utformat ett gränssnitt för att enkelt kunna använda Soar som en resonerande del i CGF utveckling (se avsnitt 4.2).

Psykologisk del

Den psykologiska delen är den del som påverkar modellen utifrån psykologiska data. I denna del ingår mekanismer som mental arbetsbelastning och situationsmedvetenhet. Ett förslag på implementation är att utnyttja Lisrel-modeller för att modellera relationer mellan sådana psykologiska faktorer. Det finns även modeller att hämta från sk Performance Moderator Functions, PMF's.

Fysisk del, sensorer och aktuatorer

Den fysiska delen innehåller modeller av fysiska faktorer som begränsad rörlighet, reaktionsförmåga, synfält, utmattning mm. Även här finns möjlighet att använda PMF's med inriktning på fysisk förmåga.

5. Beskrivning av andra projekt

I detta avsnitt beskrivs kort några nationella och internationella tidigare och nu pågående projekt.

5.1. Tidigare projekt på FOA och FFA

Vid FFA drevs tidigare ett par projekt med namnen H-pilot, E-pilot och Automatisk radarjaktledare. I H-pilot och E-pilot projekten utvecklades algoritmer för beteende i en närluftstrid. H-pilot och E-pilot fanns integrerade i simulatorn FOSIM och var där svåra att besegra. Detta var dock en följd av att den syntetiska piloten var utrustad med en väldigt orealistisk förmåga att reagera så fort den mänskliga pilotens tipp och rollvinklar förändrades.

Dessa projekt var avslutade 1995. CGF-projektet har studerat källkoden till delar av de beteenden som H-pilot kunde uppvisa. Både H-pilot och E-pilot har problem med skalbarhet i problemlösningsförmågan eftersom det inte fanns någon generell kognitiv modell bakom beteendet.

Olika typer av AI-tekniker för kunskapsrepresentation och problemlösning som t.ex. neurala nät och genetiska algoritmer används inom många projekt på FOA/FOI. I dessa projekt används de dock ej för att modellera mänskligt beteende på det sätt som det används för CGF implementation.

FOI-projektet ”Samverkande robotar” har också AI-problem med två agenter som skall samverka men här finns inget behov av att efterlikna mänsklig kognition.

5.2. Pågående och tidigare utländska projekt

5.2.1. AMBR

AMBR projektet (Agent-based Modeling and Behavioral Representation) är ett mycket stort pågående forskningsprogram, där flera olika arkitekturers (EPIC-Soar, ACT-R, iGen/COGNET, D-COG och D-OMAR) förmåga att modellera mänskligt beteende i ett enkelt flygledningsscenario jämförs. De preliminära resultaten i projektet visar att alla arkitekturer kan användas för att modellera det mänskligt beteendet men de olika problemen i modelleringsuppgiften kräver olika lösningar av varje arkitektur. Utvärdering visar att arkitekturerna uppför sig korrekt förutom i de lägen då den mänskliga responsen är oväntad eller svår att förutsäga. Projektets första fas redovisades vid ”10th Conference on Computer Generated Forces and Behavioral Representation”. Projektet forsätter under 2002-2003.

I detta amerikanska projekt har också ingått att anpassa de fem arkitekturerna till HLA och för detta har ICARUS federationen utvecklats. Alla deltagande arkitekturer är i dagsläget HLA-kompatibla.

5.2.2. JSAF och ONESAF

JSAF eller JointSAF och OneSAF är nu pågående projekt som avser att skapa en gemensam kunskapsrepresentationsarkitektur och simuleringsramverk för alla typer av entiteter som amerikanska försvarsmakten behöver simulera. Skillnaden mellan OneSAF och JSAF är inte känd av projektgruppen. Av döma av bidragen i konferensbidragen till 10th Conference on Computer Generated Forces and Behavorial Representation går dock trenden mot agentteknologi och kognitiv modellering då ModSAF:s kunskapsrepresentation visat sig vara bristfällig.

5.2.3. Tac-Air-Soar

Tac-Air-Soar är namnet på den tillämpning som deltagit i de stora amerikanska övningarna STOW'97, Coyote och Roadrunner'98. Regeldatabasen består i dagsläget av ca 7000 Soar regler och Tac-Air-Soar sägs kunna flyga alla typer av uppdrag som amerikanska flygvapnet regelmässigt flyger. I ovan nämnda övningar har hundratals agenter deltagit i simuleringarna där även bemannade simulatorer deltagit. För att kunna deltaga i de simuleringsmiljöer som använts har Soar här varit kopplat till ModSAF.

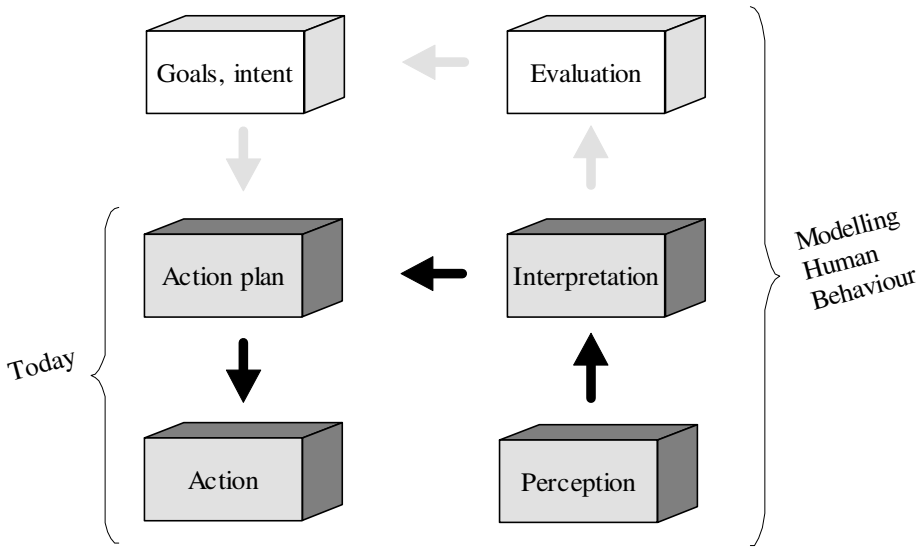
5.2.4. CHRIS

CHRIS är ett annat amerikanskt projekt som befann sig i ett uppstartsskede under 2001. Inspirerat av SEDRIS-projektets försök till standardisering är CHRIS ett försök att utveckla en standard för "human behavior representation".

5.3. Intryck från konferenser

Under "11th Conference on Computer Generated Forces and Behaviour Representations" i Orlando, Florida påtalades ett ökande behov av modellering av mänskligt beteende till skillnad från mer allmänna modeller av artificiellt beteende.
För att understryka vikten av beteendevetenskap inom området datorgenererade styrkor så har man valt att från och med nästa år kalla konferensen "Behaviour representation in modelling and simulation".

Konferensen inleddes med en syn på var området befinner sig idag och vad som kan förväntas i en närliggande framtid. Det påpekades att för att uppnå mer kompletta modeller bör man skapa beteende som verkligen förstår och inte modeller som enbart ser ut att förstå. I modellerandet av mänskligt beteende saknas idag väl utformade modeller av utvärdering, målstruktur, och intentioner. Ofta kortsluts kedjan på ett tidigare stadium. Modeller för effektiv respons mot oväntade händelser var bland annat något som borde adresseras för att uppnå mer adaptivt beteende.



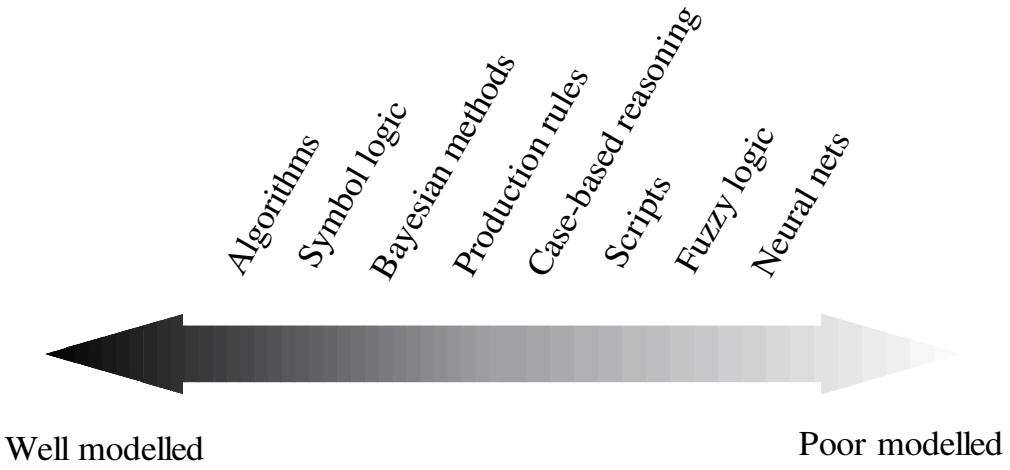
Figur 12. Kortsluten kedja vid modellering av beteende

Ett tecken på att intresset expanderat var att konferensbidragen ökat med över 70%, vilket inneburit att kvaliteten på accepterade bidrag också höjts. Trots detta kunde man identifiera några viktiga delområden som var underrepresenterade:

- ”Operator interfaces”
- ”Network interoperability”
- Terräng representation
- Tidsrepresentation
- Fysikaliska modeller
- Grundläggande kognitiv forskning

Istället handlade ett flertal bidrag om rena mjukvaruintressen, dvs arkitekturfrågor som standardiseringar, återanvändbara modeller och komponentbaserade system. Konferensbidragen var av varierande kvalitet, men ett flertal visade sig vara helt i linje med detta FOI-projekts aktiviteter, främst de bidrag som rörde beteendevetenskap, VV&A samt bibliotheksstrukturer med återanvändbara beteendekomponenter.

Vad gäller kognitiva arkitekturer så representerades system baserade på ”production rules” (regler), script och case-based reasoning. Dessa system anses ligga i mitten på en skala över hur väl kognitiva system är modellerade idag.



Figur 13. Kunskap om olika modelleringsansatser idag.

6. Diskussion

6.1. Slutsatser

Förstudien identifierade ett antal behov och frågor som måste bearbetas för att projektets långsiktiga mål skall kunna uppnås, nämligen att kunna bidra med riktlinjer för hur ett försvarsmaktsgemensamt referensbibliotek för kognitiva modeller skall se ut,:

- Vilka avvägningar mellan modellkomplexitet och utvecklingskostnad skall göras. Dessa avvägningar kommer att vara olika beroende på CGF:ernas användningsområde.
- Det kommer behöva upprättas en ontologi, d.v.s. en beskrivning av alla objekt och företeelser agenterna skall kunna resonera om, för varje domän som skall modelleras.
- Det kommer behövas en standardiserad metod för hur uppgiftsanalyser skall gå till.
- Det kommer behöva finnas ett metaspråk för referensbiblioteket, jfr CHRIS under avsnitt 5.2.4.

Dessutom ser vi ett behov av gemensamma scenarion inom pågående simuleringsprojekt inom FOI, Försvarsmakten och industrin. Det skulle då vara enklare att identifiera behoven och utveckla CGF:er för specifika syften som kan testas och utvärderas av olika användare. En tänkbar lösning vore att använda sig av intressanta scenarion från LedsystT-projektet.

Projektgruppen har inte i planerad utsträckning hunnit infoga de avancerade beteendevetenskapliga modeller som beskrivs i t.ex. Svensson mfl (1997).

Projektet har påbörjat VV&A processen, men eftersom riktlinjer för utvärdering av CGF:er i dagsläget ej är utvecklade och projektet hittills ej kunnat samköra agenterna med andra återstår mycket VV&A arbete.

6.2. Framtida inriktning av verksamheten

Följande tänkbara verksamheter har identifierats av projektgruppen inför fortsatt arbete. Alla här identifierade områden kommer dock inte att kunna behandlas under 2003-2004.

6.2.1. Forsätta utvecklingen av projektets datorgenererade flygförare

Den agent som implementerats bör tillföras mer beteende och förmåga att kunna agera i fler situationer och roller, som t ex jaktuppdrag. Projektet bör även utveckla samverkande agenter som agerar tillsammans med flygstridsledare, i rote och i fyrgrupp.

För att kunna göra detta krävs:

- Beskrivningar av typfall/scenarion.
- Utökade uppgiftsanalyser.
- Vidareutvecklad metodik för att utvinna flygförarnas faktiska beteende ur loggar (FLSC och UTA).
- Utveckla metodik för att beskriva föreskrivet beteende utifrån doktrindokument och expertintervjuer.
- En ontologi där alla begrepp (och dess relationer) relevanta för en flygförare upprättas.

6.2.2. Integrera agenter i avancerade simuleringsmiljöer

Projektet bör fortsätta arbetet med att integrera agenterna i mer utvecklade simuleringsmiljöer av typen FLSC och FENIX för att kunna utvärdera agenterna.

För att kunna göra detta krävs:

- HLA gränssnitt till utvecklade agenter (inklusive nödvändiga FOM:ar och SOM:ar).
- Tillgång till simuleringsmiljöer (t.ex. FLSC, FLAMES, FENIX eller T3Sim).

6.2.3. Fortsätta integreringen av experimentella data

Projektet bör fortsätta studera hur experimentella beteendevetenskapliga data skall infogas i CGF tillämpningar. Genom samarbetet med FOI projektet Mental Belastning och Prestation har CGF-projektet tillgång till i stort sett världsunika data och modeller från flygförardomänen och detta bör användas. På MSI-institutionen finns även insamlade data av hur beslutsfattande i staber går till.

6.2.4. Fortsätta jämförelsen av arkitekturer

Projektet bör fortsätta att jämföra och beskriva arkitekturer, verktyg och ansatser som används för CGF utveckling och kognitiv modellering enligt de kriterier som beskrivs i avsnitt 2. Fler och utökade jämförelser liknande den Soar-Agent Factory jämförelse som beskrivs i avsnitt 4 bör genomföras.

Några nya arkitekturer som bör studeras är:

- VR-Forces
- ACT-R
- Neurala nät för modellering av flygförares perception

6.2.5. Arbeta vidare med VV&A frågor

Projektet bör i samarbete med VV&A projektet utveckla metodik för hur verifiering, validering och ackreditering av beteende modeller och kognitiva modeller bör gå till.

6.2.6. Utveckla riktlinjer för ett referensbibliotek för kognitiva/beteende-modeller

Målet för projektet i ett längre tidsperspektiv är att utveckla riktlinjer för upprättandet av ett försvarsmaktsgemensamt referensbibliotek för kognitiva/beteendemodeller.

För att detta skall lyckas krävs:

- En formaliserad process för ”knowledge acquisition/elicitation” och uppgiftsanalys
- Ett implementationsoberoende språk för beskrivning av modellerna (jmfr XML).

6.2.7. Implementera CGF:er i andra domäner

De olika modelleringsarkitekturernas lämplighet för att implementera CGF i andra domäner bör studeras. Några exempel på intressanta domäner med delvis andra problem är:

- Ledningssystem
- IT-försvar
- Marin
- Markstrid

6.2.8. Studera olika aggregationsnivåer

Hittills har en flygförare på individnivå modellerats. Vid modellering av aggregerade enheter som t.ex. modellering av en hel bataljon kommer andra problem att uppenbara sig. Projektet skulle kunna börja studera dessa aggregation/disaggregationsproblem genom att modellera i en annan domän eller försöka modellera luftkriget på en högre nivå.

6.2.9. Studera alternativa användningsområden för agentteknologi

Agentteknologi kan användas i flera andra sammanhang än för CGF-implementation och dessa kan utforskas vidare. Några intressanta exempel är:

- Mobila agenter, d.v.s. agenter som kan ge sig ut och söka information på internet, sk mobil kod.
- Agenter för beslutsstöd i t.ex. Systemlyft JAS projektet
- Agenter för informationsinsamling för markstridssoldater och andra operatörer.
- Agenter som sköter informationshantering och IT-skydd på låg nivå i datasäkerhetssammanhang.

7. Referenser

Anderson, Jan (1993). *Rules of the Mind*. Hillsdale, Erlbaum.

Borg, Stellan (1996) Evaluation of COGNET a tool for Cognitive Modeling and for building Executable User Models. HFA Rapport 1996-2.

Deutsch, Steve (1999). Presentation vid ”*Panel on Human Behavior Representation*”. AIAA Modeling and Simulation Conference, Portland, Orlando.

DMSO (1995). *Modeling and Simulation Master Plan 1995*. Defence Modeling and Simulation Office.

Downes-Martin, Stephen (1995). *A Survey of Human Behavior Representation Activities for Distributed Interactive Simulation*. SAIC Rapport.

HOBM Technology Review (2001).

Holm, Gunnar & Moradi, Farshad (1997). *Distribuerad interaktiv simulering och värderingsmodeller.* FOA Rapport 97-00496-202.

McClanahan, Ted, Feinberg, Jerry, Goalwin, Patrick, Blemberg,, Paul (2001*) Human and Organizational Behavior Modeling (HOBM) Technology Assessment*. MSIAC Rapport MS-00-0019/0028.

Newell (1990). *Unified theories of cognition.* Harvard University Press.

Kieras (1994). *A guide to GOMS Task Analysis.* University of Michigan

Kieras, D., & Meyer, D. (1994). *The EPIC architecture for modeling human information-processing: A brief introduction*. TR-94/ONR-EPIC-1. University of Michigan.

Kirwan, B. & Ainsworth, L. (1992). *A guide to task analysis*. London: Taylor & Francis

Svensson, Erland, Angelborg-Thanderz, Maud, Sjöberg, Lennart, and Olsson, Stellan (1992). Risk för informationsöverflöde? Mental arbetsbelastning och prestation vid militär flygning. FOA rapport 92-50097-5.

Svensson, Erland, Angelborg-Thanderz, Maud, Sjöberg, Lennart, and Olsson, Stellan (1997). Information complexity - mental workload and performance in combat aircraft, *Ergonomics*, 40, 362-380.

Wallin, Niklas (2001). Psycho- physiological modeling in Computer Generated Forces. Examensarbete KTH.

A. Arkitekturöversikt

Arkitekturerna står i stort beskrivna i den ordning i vilken projektdeltagarna har mest kunskap om dem. Neurala nät, genetiska algoritmer, ”case-based reasoning”, ”fuzzy systems”, expert system och finita tillståndsmaskiner är egentligen inte arkitekturer utan snarare ansatser eller paradigm för problemlösning och kunskapsrepresentation. De beskrivs dock i rapporten på samma format som arkitekturerna även om inte alla värderingskriterier är tillämpliga för att beskriva dem.

Innehåll:

7.1.1. Soar

Soar är namnet på en kognitiv modelleringsarkitektur som varit under utveckling sedan 1982. Soar stod ursprungligen för State, Operator And Result men är nu snarare att betrakta som ett egennamn. Arkitekturen utvecklades ursprungligen av Allen Newell, John Laird och Paul Rosenblum och utvecklingen fortsätter vid ett antal amerikanska universitet och företaget Soar Tech. Soar är den idag mest spridda av de kognitiva modelleringsarkitekturerna.

Soar kan enligt ”the Soar FAQ” ses på tre olika sätt.

- Ett regelbaserat programmeringsspråk för implementation av artificiell intelligens tillämpningar.
- En kognitiv teori, d.v.s. en teori som beskriver hur vi människor tänker och fattar beslut.
- En samling principer och begränsningar för kognitiva processer som bildar ett ramverk inom vilket man kan utveckla kognitiva modeller

Namn	Soar
Projektets konfidens i bedömningen	Hög
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg (se dock EPIC-Soar).
Kognitivt beteende	Hög, för beteende i tidsintervallet 0,5 sek-1 h, d.v.s. inte neurokognitiva fenomen eller sociala processer.
Grupp beteende	Låg. Det finns flera projekt med samverkande entiteter, men då är en det en grupp av entiteter modellerade på individnivå. Det finns inget särskilt stöd aggregering och deaggregering men Soar bör kunna användas för att beskriva en aggregerad truppstyrka på samma sätt som en individ modelleras.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Hög, arkitekturen är mycket generell och har visat att den kan skala upp för att möta de problem man möter när man skall modellera luftkrig i stor skala genom sitt deltagande i övningarna STOW’97, Coyote och Roadrunner’98. Soar gör anspråk på att vara en så kallad ”Unified Theory of Cognition” d.v.s. en mängd experimentella data och mikroteorier kan modelleras och beskrivas i Soar.
Situatedness	OK, Soar finns t.ex. kopplat till ModSAFs flygplansmodeller i Tac-Air-Soar projektet. Frågan är dock hur detaljerade ModSAF:s flygplan är.
Deterministiskt system	Nej, om man inte uttryckligen utformar för att få det deterministiskt, d.v.s. alltid bara har en operator som kan väljas i varje situation.
Kunskapsrepresentation och beteenderepresentation	Soar är en helt symbolisk arkitektur med produktionsregler som styr de enskilda agenterna. Inferensmaskinen föreslår, väljer och tillämpar regler. Alla tillämpliga regler aktiveras parallellt och en regel väljs därefter genom ”preferens-taggning”. I långtidsminnet lagras alla regler och i arbetsminnet hanteras aktuella objekt och

	deras status. Symboler i långtidsminne och arbetsminne definieras med attribut och värde. Soar har ett målinriktat beteende och inferensmaskinen skapar automatiskt undertillstånd (”sub-goaling”) för att bryta ner komplexa problem och när modellen inte vet vad den skall göra. Inferensmaskinen kan resonera om okända attribut. Soar använder sig av rete-algoritmen för sökning i långtidsminnet.
Inlärningsförmåga eller träningsbehov	Soar har en inlärningsfunktion som skapar nya regler utifrån resultat av tidigare sub-goaling processer, d.v.s. gamla operatorer kan automatiskt kombineras, men Soar kan inte lära sig helt nya beteenden.
Övrigt	I och med Soar-Speak har Soar kopplingar till ett naturligt språkgränssnitt, d.v.s. det går att ge agenterna talade kommandon (givetvis med en begränsad vokabulär och syntaktisk förståelse men tillräckligt för flygstridsscenarion).
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja, Soar har utvecklats sedan 1982 och en mängd rapporter har publicerats.
Status på utvecklad mjukvara	Soar finns nu i version 8.4. Vidareutveckling fortgår kontinuerligt.
Mjukvaran spridd	Ja, många forskargrupper i välden använder Soar.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	I och med AMBR projektet (se avsnitt 5.1) har Soar fått ett HLA gränssnitt. Soar har genom DIS-protokollet länge kunna deltaga i större distribuerade simuleringar med hundratals agenter och bemannade simulatorer över flera kontinenter.
VVA-processen	Regelbaserade system med ”production rules” är ”enkla” att VVA:a. Soar är dock inte deterministiskt.
Modellbyggarens gränssnitt	Mycket bristfälligt. En grafisk utvecklingsmiljö vid namn VisualSoar finns men lämnar mycket i övrigt att önska.
Återanvändbarhet/ utbyggbarhet	Hög återanvändbarhet.
Integrerbart	Ja, Soar har integrerats med MODSAF, STAGE, JSAF , kommersiella spel osv
Direkta prestationsmått	
Beräkningskapacitet	Soar:s inferensmaskin kan idag producera 3-20 beslut/sekund. Målet ligger på 20 per sekund och därefter skall bara antalet agenter utökas. Rete-algoritmen gör att prestanda inte försämras märkbart med ökande regeldatabaser. Tac-Air-Soar innehåller ca 7000 regler, men i experiment har upp till 100.000 regler testats utan märkbar försämring.
Övrigt	
Tillgänglighet	Grundarkitekturen gratis nedladdningsbar. Tillämpningar som Tac-Air-Soar ägs av US Air Force.
Hårdvara/mjukvara/OS	Skrivet helt i C, körbart på alla maskiner som hanterar C.
Huvudsaklig finansiär	Amerikanska försvarsdepartementet och kommersiella intresssen.
Omfattning finansiering	5 miljoner dollar 2001
Utvecklare	University of Michigan, the University of Southern California, Carnegie Mellon University, Soar Tech
Användare	Akademiska grupper (huvudsakligen i USA men även

	internationellt), US Air Force?
Användningsområde	Soar har bl.a. använts till att modellera individuella piloter och flygledare.
Tillämpningar/exempel på projekt	Tac-Air-Soar, RWA Soar (Rotary wing air-Soar)
Referenser	www.soartech.com www.umich.edu/soar Newell, 1990

7.1.2. Agent Factory

Namn	Agent Factory
Projektets konfidens i bedömningen	Hög. Projektet har arbetat med Agent Factory.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Medel
Kognitivt beteende	Medel
Grupp beteende	Medel
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Hög, dock måste en modellerare (som i många andra arkitekturer) definiera alla begrepp som agenten skall kunna resonera kring i förväg.
Situatedness	Hög
Deterministiskt system	Ja
Kunskapsrepresentation och beteenderepresentation	Inferensmaskinen är både framåt och bakåtlänkande, detta innebär att den kan resonera sig fram till slutsatser utifrån en given situation men också resonera om vad det saknar för att kunna lösa situationen. Agentens beteende beskrivs i ett Mål-plan-fas-träd. För mer information om kunskapsrepresentationen se avsnitt 4.1.
Inlärningsförmåga eller träningsbehov	Nej.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Två artiklar som beskriver idéerna bakom arkitekturen finns publicerade.
Status på utvecklad mjukvara	Version 1 finns utvecklad, version 2 är under utveckling.
Mjukvaran spridd	Nej.
Används i simuleringar	En enkel demo finns..
Används i distribuerade simuleringar	Det har ej visats, men eftersom Agent Factory bygger på HLA kommer det inte vara några problem.
VVA-processen	-
Modellbyggarens gränssnitt	Agent Factory har ett väl utbyggd grafiskt gränssnitt avsett att underlätta för domänexperter, d.v.s. inte programmerare att deltaga

	i modellutvecklingen.
Återanvändbarhet/ utbyggbarhet	Hög.
Integrerbart	Bör vara möjligt att integrera med andra HLA kompatibla tillämpningar.
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	I ett utvecklingsstadium.
Hårdvara/mjukvara/OS	COTS.
Huvudsaklig finansiär	FMV.
Omfattning finansiering	?
Utvecklare	Pitch Kunskapsutveckling.
Användare	-
Användningsområde	Verktyg för att utveckla agenter till många olika typer av domäner.
Tillämpningar/exempel på projekt	
Referenser	www.pitch.se

7.1.3. ACT-*

Namn	ACT-R, ACT-PM Adaptive Control of Thought Rules eller Perceptual Motor
Projektets konfidens i bedömningen	Medel.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Hög.
Kognitivt beteende	Hög.
Grupp beteende	Låg, det finns inget särskilt stöd för modellering av gruppbeteende då arkitekturen är avsedd för modellering av enskilda individers kognition.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Kan troligen skalas upp för att möta problemen i komplexa simuleringar men det har ej demonstrerats. ACT-R gör anspråk på att vara en ”Unified Theory of Cognition”.
Situatedness	Troligen OK.
Deterministiskt system	Nej.
Kunskapsrepresentation och beteenderepresentation	ACT-R är ett exempel på en hybrid arkitektur där både symbolisk kunskapsrepresentation (d..vs. production rules, if-then satser) och subsymboliska regler (d.v.s. neurala nät) används. Den subsymboliska aktivationen används för att avgöra vilken regel som skall väljas när det finns flera tillämpliga regler. I ACT-*

	skiljer man mellan deklarativ kunskap (d.v.s. att veta att) och procedurell kunskap (d.v.s. att veta hur).
Inlärningsförmåga eller träningsbehov	ACT-* innehåller funktioner för inlärning och genom hur element i arbetsminnet aktiveras via de subsymboliska reglerna kan nya beteenden läras in. Att ställa in de subsymboliska parametrarna kan dock vara svårt och omständligt.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja, en stor mängd publikationer finns.
Status på utvecklad mjukvara	Mjukvaran har utvecklats i flera versioner under åren. Senaste version 5.0 finns nu tillgänglig i beta version.
Mjukvaran spridd	Ja, ACT-R används av andra än utvecklarna.
Används i simuleringar	Oklart om ACT-* används i något tillräckligt komplext scenario.
Används i distribuerade simuleringar	Ja, men det är först i och med AMBR projektet (se avsnitt 5.1) som en HLA koppling utvecklats
VVA-processen	Den subsymboliska aktivationen kan vara svår att hantera i VV&A processen.
Modellbyggarens gränssnitt	Inbyggt i ACT-R finns ett GUI, Grafiska användargränssnitt, som kallas AGI. AGI:et underlättar utvecklingen i ACT-R och kan också användas till debugging och testning. AGI:et lämpar sig ej för uppbyggnad av större komplexa system utan främst för utveckling av mindre system. Kod som utvecklas i AGI:et är portabel mellan olika system. Utveckling av program kan också ske i något annat GUI i LISP. ACT-R har också en egen syntax för att skriva regler som modelleraren/utvecklaren måste lära sig.
Återanvändbarhet/ utbyggbarhet	Ingen info, men de symboliska reglerna bör gå att återanvända.
Integrerbart	Oklart om ACT-* integrerats med annan programvara förrän i AMBR projektet.
Direkta prestationsmått	
Beräkningskapacitet	Ingen info
Övrigt	
Tillgänglighet	Grundarkitekturen är fritt nedladdningsbar.
Hårdvara/mjukvara/OS	ACT-* är skrivet i LISP och för MacIntosh, men finns nu för PC.
Huvudsaklig finansiär	Amerikanska försvarsdepartementet
Omfattning finansiering	?
Utvecklare	Carnegie Mellon University
Användare	Carnegie Mellon University, USAF Research Labs, US Army Research Labs
Användningsområde	Modellering av individers kognition, språkförståelse o.s.v.
Tillämpningar/exempel på projekt	-
Referenser	http://act.psy.cmu.edu/ACT/act-home.html Anderson, 1993

7.1.4. EPIC

Namn	EPIC (Executive-Process Interactive Control)
Projektets konfidens i bedömningen	Låg
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Hög
Kognitivt beteende	Hög
Grupp beteende	Låg
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Begränsad skalbarhet. EPIC är framför allt starkt när man skall modellera perception.
Situatedness	Hög, arkitekturen har utvecklas för att modellera perceptuellt och motorisk beteende när en människa t.ex. tittar på skärmar och trycker på knappar.
Deterministiskt system	?
Kunskapsrepresentation och beteenderepresentation	”Production rules” skrivna i LISP. Ingen särskilt hantering av mål utan de hanteras som vilket arbetsminneselement som helst.
Inlärningsförmåga eller träningsbehov	Nej
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja.
Status på utvecklad mjukvara	?
Mjukvaran spridd	I begränsad omfattning.
Används i simuleringar	Framför allt med i och med EPIC-Soar. EPIC är framför allt bra på att modellera perception
Används i distribuerade simuleringar	Nej, men EPIC-Soar är med i AMBR projektet.
VVA-processen	?
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	Oklart, men går uppenbarligen att integrera med Soar och Soar har integrerats med många andra verktyg (se EPIC-Soar).
Direkta prestationsmått	
Beräkningskapacitet	
Övrigt	
Tillgänglighet	Oklart, men grundarkitekturen är troligen fritt nedladdningsbar.

Hårdvara/mjukvara/OS	EPIC är skrivet i LISP.
Huvudsaklig finansiär	Amerikanska försvarsdepartementet.
Omfattning finansiering	?
Utvecklare	University of Michigan.
Användare	Air Force Research Labs, DMSO.
Användningsområde/	Människa maskin interaktionsstudier.
Tillämpningar/exempel på projekt	
Referenser	Kieras & Meyer, 1994 McClanahan mfl, 2001

7.1.5. EPIC-Soar

Arkitekturen EPIC och arkitekturen Soar har i EPIC-Soar integrerats då EPIC är lämplig om man vill ha detaljerade perceptionsmodeller medan Soar är starkare när beslutsfattande skall modelleras. EPIC-Soar ärver egenskaperna från de två arkitekturerna.

Namn	EPIC-Soar
Projektets konfidens i bedömningen	Medel.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Hög.
Kognitivt beteende	Hög.
Grupp beteende	Låg, inget särskilt stöd för att modellera grupper. Grupper modelleras som enskilda individer.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Ja.
Situatedness	Mycket bra.
Deterministiskt system	Beror på utformningen av Soar reglerna.
Kunskapsrepresentation och beteenderepresentation	Samma som i Soar.
Inlärningsförmåga eller träningsbehov	Samma som i Soar.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Några publikationer och en doktorsavhandling.
Status på utvecklad mjukvara	?
Mjukvaran spridd	Nej.
Används i simuleringar	Används i AMBR projektet.
Används i distribuerade	Används i AMBR projektet.

simuleringar	
VVA-processen	-
Modellbyggarens gränssnitt	Inget avancerat stöd för modellbyggaren finns utvecklat.
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	3-20 agenter
Övrigt	
Tillgänglighet	Gratis nedladdningsbart
Hårdvara/mjukvara/OS	Linux
Huvudsaklig finansiär	100.000 dollar.
Omfattning finansiering	Amerikanska försvarsdepartementet.
Utvecklare	University of Michigan.
Användare	?
Användningsområde	-
Tillämpningar/exempel på projekt	AMBR projektet.
Referenser	McClanahan, 2001

7.1.6. D-COG

Namn	D-COG (Distributed Cogniton)
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Medel.
Kognitivt beteende	Hög.
Grupp beteende	Låg.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Troligen tillfredställande, men det har inte demonstrerats.
Situatedness	?
Deterministiskt system	?
Kunskapsrepresentation och beteenderepresentation	?
Inlärningsförmåga eller träningsbehov	?
Övrigt	-

Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja, men inte i särskilt stor omfattning.
Status på utvecklad mjukvara	Fortfarande i version 1.0.
Mjukvaran spridd	Nej.
Används i simuleringar	Oklart.
Används i distribuerade simuleringar	D-COG är med som en arkitekturer i AMBR projektet men förutom det är det oklart.
VVA-processen	?
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	GOTS när det är färdigutvecklat. Finns inte tillgängligt ännu.
Hårdvara/mjukvara/OS	COTS.
Huvudsaklig finansiär	Amerikanska försvarsdepartementet.
Omfattning finansiering	?
Utvecklare	USAF Research Labs
Användare	?
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	McClanahan mfl, 2001

7.1.7. D-OMAR

Namn	D-OMAR (Distributed Operator Model Architecture)
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Medel.
Kognitivt beteende	Hög.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Inte helt säkert att arkitekturen klarar att skala upp.
Situatedness	?
Deterministiskt system	?
Kunskapsrepresentation och beteenderepresentation	?
Inlärningsförmåga eller träningsbehov	?
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Det finns några publikationer.
Status på utvecklad mjukvara	?
Mjukvaran spridd	Verkar inte vara spridd.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Ja, i AMBR-projektet.
VVA-processen	?
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	GOTS.
Hårdvara/mjukvara/OS	Fungerar inte i Windows miljö, portningsarbete pågår. Byggt med JAVA, C++ och LISP.
Huvudsaklig finansiär	Amerikanska försvarsdepartementet.

Omfattning finansiering	?
Utvecklare	BBN Inc.
Användare	USAF Research Labs.
Användningsområde	?
Tillämpningar/exempel på projekt	?
Referenser	Deutsch, 1999 McClanahan, 2001

7.1.8. iGen/COGNET

Namn	iGen är den kommersiella tillämpningen av det tidigare akademiska COGNET (Cognition as a Network of Tasks).
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Medel.
Kognitivt beteende	Hög.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Hög.
Situatedness	?
Deterministiskt system	?
Kunskapsrepresentation och beteenderepresentation	Beteenden byggs upp av nätverk av handlingar (tasks) där komplexa beteenden byggs upp av mindre beteenden.
Inlärningsförmåga eller träningsbehov	Nej.
Övrigt	
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja, men det är huvudsakligen utvecklarna som publicerat något.
Status på utvecklad mjukvara	I och med iGen finns arkitekturen som kommersiell programvara.
Mjukvaran spridd	Ja.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Ja.
VVA-processen	?
Modellbyggarens gränssnitt	?
Återanvändbarhet/	?

utbyggbarhet	
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	COTS, kostar ett par hundratusen kronor.
Hårdvara/mjukvara/OS	COTS.
Huvudsaklig finansiär	Amerikanska försvarsdepartementet och kommersiella intressen.
Omfattning finansiering	
Utvecklare	CHI Systems.
Användare	CHI Systems, enligt reklamen använder SAAB och BaE systems också COGNET.
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	www.cognitiveagent.com McClanahan mfl, 2001 Borg, Stellan, 1996

7.1.9. MICROSAINT

Namn	MicroSaint
Projektets konfidens i bedömningen	Hög.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Medel. MicroSaint har inget direkt stöd för någon kognitiv modellering utan är snarare att betrakta som ett verktyg för att skapa körbara flödesschema.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Ja, troligen tillräcklig.
Situatedness	OK, men upp till modelleraren.
Deterministiskt system	Ja.
Kunskapsrepresentation och beteenderepresentation	Med MicroSaint definierar modelleraren flödesschema som sedan ör körbara. Kunskapsrepresentationem sker genom dessa flödesscheman.
Inlärningsförmåga eller träningsbehov	Nej.
Övrigt	-
Ansatsens/arkitekturens	

mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja, men MicroSaint används framför allt i industrin.
Status på utvecklad mjukvara	Kommersiell mjukvaruprodukt, kostar ca 100.000 kr.
Mjukvaran spridd	Ja.
Används i simuleringar	Ja, se IMPRINT.
Används i distribuerade simuleringar	Tveksamt.
VVA-processen	MicroSaint modeller är tämligen rättframma att analysera i VV&A processen.
Modellbyggarens gränssnitt	MicroSaint har en grafisk utvecklingsmiljö som innehåller visst stöd för datainsamling och visualisering.
Återanvändbarhet/ utbyggbarhet	Ja, MicroSaint flödesschema går att återanvända och bygga ut. Dock inget riktigt modulärt tänkande.
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	COTS.
Hårdvara/mjukvara/OS	COTS.
Huvudsaklig finansiär	Kommersiella intressen.
Omfattning finansiering	?
Utvecklare	Micro Analysis and Design.
Användare	Många användare.
Användningsområde	Design av människa maskin gränssnitt.
Tillämpningar/exempel på projekt	Har framför allt används i utformning av operatörsplatser till tex Comanchehelikoptern.
Referenser	www.mad.com

7.1.10. IMPRINT

Namn	IMPRINT
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Medel.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	IMPRINT är anpassat till flygoperationer, d.v.s. till en särskild nisch så skalbarheten är eventuellt begränsad.
Situatedness	?

Deterministiskt system	?
Kunskapsrepresentation och beteenderepresentation	IMRINT är baserat på modelleringsverktyget MicroSaint
Inlärningsförmåga eller träningsbehov	Troligen inte.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja, men inte i särskilt stor utsträckning.
Status på utvecklad mjukvara	Oklart.
Mjukvaran spridd	Baserat på MicroSaint som är mycket spritt. IMPRINT i sig har troligen ingen större spridning.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Oklart, men IMPRINT är HLA-kompatibelt.
VVA-processen	?
Modellbyggarens gränssnitt	Antagligen samma gränssnitt som i MicroSaint där det finns ett väl utvecklat grafiskt gränssnitt.
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	GOTS
Hårdvara/mjukvara/OS	COTS
Huvudsaklig finansiär	Amerikanska försvarsdepartementet
Omfattning finansiering	7 miljoner dollar
Utvecklare	US Army Research Labs, USAF Research Labs utvecklar CART
Användare	?
Användningsområde	Människa System Integration.
Tillämpningar/exempel på projekt	CART, Air Warrior
Referenser	McClanahan mfl, 2001

7.1.11. MIDAS

Namn	MIDAS (Man-machine Integrated Design and Analysis System).
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt	Medel.

beteende	
Kognitivt beteende	Hög.
Grupp beteende	Medel, visst stöd för modellering av besättningar och befälhavare finns.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	OK.
Situatedness	Hög.
Deterministiskt system	?
Kunskapsrepresentation och beteenderepresentation	?
Inlärningsförmåga eller träningsbehov	?
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Några publikationer finns, men MIDAS befinner sig på ett forsknings stadium.
Status på utvecklad mjukvara	På ett forskningsstadium men mjukvara finns utvecklad.
Mjukvaran spridd	I viss utsträckning.
Används i simuleringar	Nej, troligen inte.
Används i distribuerade simuleringar	Nej, t.ex. inga försök till HLA anpassning.
VVA-processen	?
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	GOTS.
Hårdvara/mjukvara/OS	COTS.
Huvudsaklig finansiär	Amerikanska försvarsdepartementet och kommersiella intressen.
Omfattning finansiering	?
Utvecklare	NASA.
Användare	NASA.
Användningsområde	Används för att modellera människa maskin interaktion.
Tillämpningar/exempel på projekt	
Referenser	McClanahan, 2001

7.1.12. ModSAF inkl OneSAF och JSAF

Namn	ModSAF (Modular Semi Automated Forces) samt efterföljarna One Semi Automated Force och Joint Semi Automated Forces.
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Medel.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Antalet tillstånd och tillståndsövergångar växer mycket snabbt vid komplexa beteenden och det kan bli ohanterligt.
Situatedness	Begränsad.
Deterministiskt system	Nej.
Kunskapsrepresentation och beteenderepresentation	Olika typer av finita tillståndsmaskiner.
Inlärningsförmåga eller träningsbehov	Nej.
Övrigt	Åtminstone i ModSAF behövs det mänskliga operatörer som tar taktiska beslut på högre nivå, d.v.s. agenterna är inte autonoma och måste hela tiden övervakas
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja.
Status på utvecklad mjukvara	ModSAF har funnits relativt länge och måste betraktas som moget. Utveckling i OneSAF och JSAF pågår.
Mjukvaran spridd	Ja, inom amerikanska försvaret.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Ja.
VVA-processen	Finita tillståndsmaskiner är relativt rättframma att validera så länge det inte är komplexa beteenden som skall modelleras.
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	Ja, delar av tillståndsmaskinerna bör definitivt gå att återanvända, men när de skall byggas ut eller modifieras kan det bli omständligt med nya tillståndsövergångar.
Integrerbart	Ja, har integrerats med flera olika simulatormiljöer.
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	

Tillgänglighet	Ej tillgänglig utanför det amerikanska försvaret.
Hårdvara/mjukvara/OS	?
Huvudsaklig finansiär	Amerikanska försvarsdepartementet.
Omfattning finansiering	?
Utvecklare	?
Användare	Amerikanska försvaret
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	-

7.1.13. STRIVE

Namn	STRIVE (Synthetic Tactical Real-time Interactive Virtual Environment)
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Låg.
Grupp beteende	Låg?
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	?
Situatedness	?
Deterministiskt system	
Kunskapsrepresentation och beteenderepresentation	
Inlärningsförmåga eller träningsbehov	
Övrigt	
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Nej
Status på utvecklad mjukvara	Projektet har kontaktat CAE försäljningsavdelningen för att få mer info men har ännu inte fått något svar.
Mjukvaran spridd	?
Används i simuleringar	?
Används i distribuerade simuleringar	?
VVA-processen	?
Modellbyggarens gränssnitt	

Återanvändbarhet/ utbyggbarhet	
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	Kommersiell programvara
Hårdvara/mjukvara/OS	?
Huvudsaklig finansiär	Kommersiella intressen.
Omfattning finansiering	?
Utvecklare	CAE
Användare	?
Användningsområde	?
Tillämpningar/exempel på projekt	?
Referenser	http://www.cae.com/en/military/software/strive.shtml

7.1.14. VR Forces

Namn	VR Forces
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Låg.
Grupp beteende	Medel, det sägs finnas stöd för aggregering t.ex. kan man ge order på högre nivåer som sedan via simulerad kommunikation sänds nedåt i kommandokedjan.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	?
Situatedness	?
Deterministiskt system	Ja, troligen.
Kunskapsrepresentation och beteenderepresentation	Det finns ett antal handingar (t.ex. förflytta sig eller skjuta) som med hjälp av if-then satser kan bygga upp mer komplexa beteenden.
Inlärningsförmåga eller träningsbehov	
Övrigt	Simulatorinstruktörer kan gå in och styra de syntetiska entiteterna i realtid, t.ex. ge order på hög nivå vilka sedan rapporteras nedåt. Oklart om detta innebär att en operatör måste sitta med.
Ansatsens/arkitekturens	

mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	?
Status på utvecklad mjukvara	Första versionen av mjukvaran släpptes 1999.
Mjukvaran spridd	?
Används i simuleringar	?
Används i distribuerade simuleringar	?
VVA-processen	?
Modellbyggarens gränssnitt	Det finns ett grafiskt användargränssnitt för att underlätta för modellbyggaren och ett objekt orienterat gränssnitt för programmerarna (API). De syntetiska entiteternas beteende (t.ex. hur långt de ser) styrs delvis genom parametersättning, d.v.s. ingen avancerad programmering behövs.
Återanvändbarhet/ utbyggbarhet	Planer för plattformar kan återanvändas.
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	Kommersiell mjukvara, kostar ca 400.000 kr.
Hårdvara/mjukvara/OS	?
Huvudsaklig finansiär	Kommersiella intressen.
Omfattning finansiering	?
Utvecklare	MäK Technologies,
Användare	?
Användningsområde	?
Tillämpningar/exempel på projekt	?
Referenser	www.mak.com/vr-forces.htm

7.1.15. TACSI

Namn	TACSI (TACtical Simulation)
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Låg.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i	Tillräcklig, alla lösningar som baserar sig på finita tillståndsmaskiner

problemlösningsförmågan	kan dock vara alltför begränsade för riktigt komplexa beteenden/situationer.
Situatedness	Inte i fokus.
Deterministiskt system	Ja.
Kunskapsrepresentation och beteenderepresentation	Hierarkiska finita tillståndsmaskiner?
Inlärningsförmåga eller träningsbehov	Nej, har inte bedömts vara intressant.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Nej.
Status på utvecklad mjukvara	?
Mjukvaran spridd	Tacsi används internt på SAAB och har integrerats i FLSC.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Ja.
VVA-processen	?
Modellbyggarens gränssnitt	Det finns ett grafiskt gränssnitt, domänexperter (icke-programmerare) kan bygga beteenden.
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	Ägs av SAAB.
Hårdvara/mjukvara/OS	?
Huvudsaklig finansiär	SAAB
Omfattning finansiering	?
Utvecklare	SAAB.
Användare	SAAB.
Användningsområde	Scenariogenereringsverktyg i SAAB:s flygsimuleringar.
Tillämpningar/exempel på projekt	FOX 1, PMSIM
Referenser	

7.1.16. STAGE

STAGE är scenariogenereringsverktyg med vissa funktioner för att hantera datorgenererade styrkor. Scriptspråket som styr de simulerade entiteternas beteende är dock relativt begränsat.

Namn	STAGE, (Scenario Toolkit And Generation Environment)
Projektets konfidens i	Medel.

bedömningen	
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Låg.
Grupp beteende	Låg.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Mycket begränsad.
Situatedness	Inte i fokus.
Deterministiskt system	Ja
Kunskapsrepresentation och beteenderepresentation	Beteenden definieras med hjälp av STAGE:s scriptspråk vilket bara räcker för relativt enkla beteenden. Sk ”user models” kan läggas till för att utveckla beteendet.
Inlärningsförmåga eller träningsbehov	Nej.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Nej. Det finns inga anspråk på att vara en arkitektur för kognitiv modellering.
Status på utvecklad mjukvara	Kommersiell mjukvara.
Mjukvaran spridd	Ja.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Ja.
VVA-processen	-
Modellbyggarens gränssnitt	Det är omständligt att definiera scenarion och beteende med STAGE.
Återanvändbarhet/ utbyggbarhet	-
Integrerbart	Ja.
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	Kommersiell programvara.
Hårdvara/mjukvara/OS	PC, UNIX.
Huvudsaklig finansiär	Kommersiella intressen.
Omfattning finansiering	?
Utvecklare	Virtual Prototypes
Användare	STAGE används t.ex. i FLSC.
Användningsområde	Scenariogenereringsverktyg
Tillämpningar/exempel på projekt	-
Referenser	

7.1.17. FLAMES

FLAMES är ett scenariogenereringsverktyg med viss funktionalitet för att hantera datorgenerade styrkor. FLAMES är egentligen ett samlingsnamn för ett antal mjukvarumoduler som behövs för scenariogenerering och omvärldshantering.

Namn	FLAMES (Flexible Analysis and Mission Effectiveness System)
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Låg.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Begränsad.
Situatedness	Inte i fokus.
Deterministiskt system	Ja.
Kunskapsrepresentation och beteenderepresentation	Scriptbaserad språk används för att definiera beteendet.
Inlärningsförmåga eller träningsbehov	Nej.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Nej.
Status på utvecklad mjukvara	Kommersiell mjukvara. Version 4.4 släpptes i september 2001.
Mjukvaran spridd	Ja.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Ja.
VVA-processen	-
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	Scripten i de moduler som i FLAMES kallas ”kognitiva modeller” kan återanvändas.
Integrerbart	
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	

Tillgänglighet	Kommersiell programvara.
Hårdvara/mjukvara/OS	IRIX, Solaris, Windows 2000.
Huvudsaklig finansiär	Kommersiella intressen.
Omfattning finansiering	?
Utvecklare	Ternion
Användare	Amerikanskt försvar och företag. I Sverige används FLAMES bl.a av FOI och FMV.
Användningsområde	Scenariogenerering.
Tillämpningar/exempel på projekt	-
Referenser	

7.1.18. Neurala nät

Artificiella neurala nät (ANN) är en kunskapsrepresentations och problemlösnings-ansats som inspirerats av hur den mänskliga hjärnan är uppbyggd på lägsta nivå, d.v.s. en stor mängd enkla neuroner som är sammankopplade och som tillsammans klarar av att lösa mer sammansatta problem. ANN representerar ett eget stort delområde inom AI forskningen.

Namn	Artificiella neurala nät (ANN)
Projektets konfidens i bedömningen	Hög.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Hög.
Kognitivt beteende	Medel.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Om man använder sig av rena ANN lösningar kommer det att bli mycket oöverskådligt att modellera t.ex. ett komplext luftstridsscenario.
Situatedness	Inte i fokus.
Deterministiskt system	Generellt sett nej, men det är avhängigt träningsdatan och om nätet övertränas eller ej.
Kunskapsrepresentation och beteenderepresentation	Neurala nät sägs av en av ANN-pionjärerna vara det nästa bästa sättet att lösa ganska många problem, d.v.s. om man inte vet tillräckligt mycket om det modellerade fenomenet för att kunna definiera formella regler kan neurala nät används för att försöka hitta reglerna som styr det. Neurala nät är bra på att hantera diffus och motsägelsefull information.
Inlärningsförmåga eller träningsbehov	Neurala nät behöver tränas för att lära sig klassificera situationer och problem i en stor mängd körningar. För bra resultat behövs en stor mängd representativ träningsdata vilket ofta kan vara svårt att få tag på. De neurala näten kan inte lära sig något som inte finns i träningsdata. De neurala nät kan lära sig klassificera stimuli på oväntade sätt och därigenom se nya vägar genom problem.
Övrigt	-

Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja, forsning kring neurala nät är nästan att betrakta som ett eget forskningsområde, med egna konferenser och tidskrifter.
Status på utvecklad mjukvara	Många olika mjukvaror finns.
Mjukvaran spridd	Ja.
Används i simuleringar	Tveksamt, se skalbarhet.
Används i distribuerade simuleringar	Nej, även om agenter uttryckta i andra formalismer skulle kunna ha moduler, för t.ex. perception, som använde sig av neurala nät.
VVA-processen	Neurala nät kommer att vara mycket svåra att granska internt eftersom beteenden styrs av vikten på noder och övergångar. Kunskapen uttrycks i subsymboliska termer vilket inte kommer säga en expert någonting. ”Principal component analysis” kan dock i viss utsträckning användas för att analysera nätens inre.
Modellbyggarens gränssnitt	Ofta inte så utvecklat. Definitionen av nätens utformning (antal noder och lager o.s.v.) är mer konst än vetenskap.
Återanvändbarhet/ utbyggbarhet	Ingen, näten måste tränas om och det finns inga delar av nätet som kan flyttas vidare till en ny agent.
Integrerbart	
Direkta prestationsmått	
Beräkningskapacitet	-
Övrigt	
Tillgänglighet	Finns fritt nedladdningsbara programvaror t.ex. en ”tool-box” till MatLab.
Hårdvara/mjukvara/OS	Inga problem.
Huvudsaklig finansiär	-
Omfattning finansiering	-
Utvecklare	-
Användare	-
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	-

7.1.19. Finita tillståndsmaskiner

Namn	Finita tillståndsmaskiner, finns i flera olika varianter. Finite State Machines (FSM) Augumented FSM Extended FSM Hierarchical FSM Non-deterministic Hierarchical FSM (Markov Models)
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Låg.
Grupp beteende	Låg.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Begränsad, antalet tillstånd och tillståndsövergångar växer väldigt fort för komplexa beteenden.
Situatedness	-
Deterministiskt system	Ja, om man inte använder fuzzy FSM:s eller markovmodeller.
Kunskapsrepresentation och beteenderepresentation	Kunskapen uttrycks som ett antal tillstånd och tillståndsövergångar.
Inlärningsförmåga eller träningsbehov	Nej.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja.
Status på utvecklad mjukvara	?
Mjukvaran spridd	?
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Ja.
VVA-processen	Finita tillståndsmaskiner är enkla att hantera i VV&A processen så länge antalet tillstånd är litet.
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	Används i många av dagens kommersiella dataspel och i ModSAF/OneSAF/JSAF och TACSI.
Direkta prestationsmått	
Beräkningskapacitet	-

Övrigt	
Tillgänglighet	-
Hårdvara/mjukvara/OS	FSM ger snabb och kompakt mjukvarukod.
Huvudsaklig finansiär	-
Omfattning finansiering	-
Utvecklare	-
Användare	-
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	

7.1.20. Genetiska algoritmer

Genetiska algoritmer är ett område som är mycket ”hett” inom AI forskningen just nu. Inspiration från naturens evolutionära utvecklingsprocesser har överförts till automatisk algoritm generering där ett antal enkla beståndsdelar korsas och muteras tills man hittat ett tillräckligt bra beteende.

Namn	Genetiska algoritmer, genetic computing
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Medel.
Kognitivt beteende	Medel.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	Genetiska algoritmer är funktionella när det handlar om att hitta t.ex. en funktion som beskriver en matematisk kurva, men det är oklart hur användbara de är för att hitta skapa beteenden i t.ex. en komplex luftstridsscenario.
Situatedness	-
Deterministiskt system	Nej, inte under evolutionen. Eventuellt därefter beroende på genernas utformning.
Kunskapsrepresentation och beteenderepresentation	Utvecklaren definierar ett visst antal enkla beteenden eller egenskaper som sedan automatiskt kombineras och korsas för att skapa nya beteenden. De mest lämpliga beteenden väljs sedan ut genom en darwinistisk utslagningsprocess. Beteenden som överlevt korsas igen o.s.v. i ett antal generationer till man hittat ett tillräckligt bra beteende.
Inlärningsförmåga eller träningsbehov	Ja.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt	Ja, men forskningen kring genetiska algoritmer befinner sig fortfarande

granskade resultat	i ett grundläggande skede.
Status på utvecklad mjukvara	?
Mjukvaran spridd	?
Används i simuleringar	Nej.
Används i distribuerade simuleringar	Nej.
VVA-processen	-
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	Gener kan återanvändas men de måste gå igenom den evolutionära processen igen då de återanvänds.
Integrerbart	
Direkta prestationsmått	
Beräkningskapacitet	-
Övrigt	
Tillgänglighet	-
Hårdvara/mjukvara/OS	-
Huvudsaklig finansiär	-
Omfattning finansiering	-
Utvecklare	-
Användare	-
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	-

7.1.21. Fuzzy Systems

Fuzzy logic eller dess användning i fuzzy systems är inte att betrakta som en modelleringsarkitektur men tas upp här för att täcka begreppet. I fuzzy logic definieras inte ett objekt egenskaper med exakta, helt skarpa gränser, t.ex. att varmt väder innebär att det är 25 grader C och allt därunder är kallt väder. Genom att gränserna är oskarpa och att t.ex. vädret kan vara lite varmt och litet kallt om det är 20 grader kan system med fuzzy logic på ett bättre sätt är ordinära logiksystem resonera om vag och oprecis information.

Namn	Fuzzy System, fuzzy logic
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Medel.
Kognitivt beteende	Medel.
Grupp beteende	Medel.
Problemlösning och	

kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	?
Situatedness	-
Deterministiskt system	-
Kunskapsrepresentation och beteenderepresentation	Objekts egenskaper definieras med suddiga gränser, t.ex. vädret är ganska varmt men litet kallt osv.
Inlärningsförmåga eller träningsbehov	-
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	-
Status på utvecklad mjukvara	-
Mjukvaran spridd	-
Används i simuleringar	-
Används i distribuerade simuleringar	-
VVA-processen	-
Modellbyggarens gränssnitt	-
Återanvändbarhet/ utbyggbarhet	-
Integrerbart	-
Direkta prestationsmått	
Beräkningskapacitet	-
Övrigt	
Tillgänglighet	-
Hårdvara/mjukvara/OS	-
Huvudsaklig finansiär	-
Omfattning finansiering	-
Utvecklare	-
Användare	-
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	-

7.1.22. Expert system

Expert system är ett sorts samlingsnamn på de tidiga regelbaserade systemen och det är i här AI forskningen har sitt ursprung. Värdering enligt kriterierna nedan är ej särskilt lämpligt men finns med för att täcka begreppet.

Namn	Expert System
Projektets konfidens i bedömningen	Låg.
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Beror på vilken data man utgår från.
Kognitivt beteende	Beror på vilken data man utgår från.
Grupp beteende	Beror på vilken data man utgår från.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	
Situatedness	-
Deterministiskt system	Ja.
Kunskapsrepresentation och beteenderepresentation	Kunskapen uttrycks som en samling ”production rules”, if-then satser.
Inlärningsförmåga eller träningsbehov	Nej.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Många publikationer.
Status på utvecklad mjukvara	Expert System måste sägas vara en mogen teknologi.
Mjukvaran spridd	Ja.
Används i simuleringar	Ja.
Används i distribuerade simuleringar	Ja.
VVA-processen	-
Modellbyggarens gränssnitt	Det finns COTS verktyg som stöder utvecklingen av expert system.
Återanvändbarhet/ utbyggbarhet	Ja, reglerna kan återanvändas men man måste bygga nya system varje gång.
Integrerbart	-
Direkta prestationsmått	
Beräkningskapacitet	-
Övrigt	

Tillgänglighet	-
Hårdvara/mjukvara/OS	-
Huvudsaklig finansiär	-
Omfattning finansiering	-
Utvecklare	-
Användare	-
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	-

7.1.23. Case-based reasoning

Case based reasoning är likt genetiska algoritmer inte en arkitektur eller verktyg likt de övriga beskrivningarna under 2.3. Case base reasoning är istället mer att betrakta som en inspirationskälla. Agenterna planerar och agerar här genom att analysera hur de gjorde i tidigare liknande situationer.

Namn	Case based reasoning
Projektets konfidens i bedömningen	Låg
Vetenskaplig grund för att modellera:	
Perceptuellt och motoriskt beteende	Låg.
Kognitivt beteende	Hög.
Grupp beteende	Medel.
Problemlösning och kunskapsrepresentation	
Skalbarhet i problemlösningsförmågan	?
Situatedness	?
Deterministiskt system	Beror på hur kunskapen representeras.
Kunskapsrepresentation och beteenderepresentation	Problemlösnigen sker genom att agenten analyserar hur den gjorde i tidigare liknande situationer.
Inlärningsförmåga eller träningsbehov	Agenten måste en omfattande databas av tidigare agerande för att kunna agera i en komplex situation.
Övrigt	-
Ansatsens/arkitekturens mogenhetsgrad	
Publicerade, vetenskapligt granskade resultat	Ja.
Status på utvecklad mjukvara	?
Mjukvaran spridd	?
Används i simuleringar	?
Används i distribuerade simuleringar	?

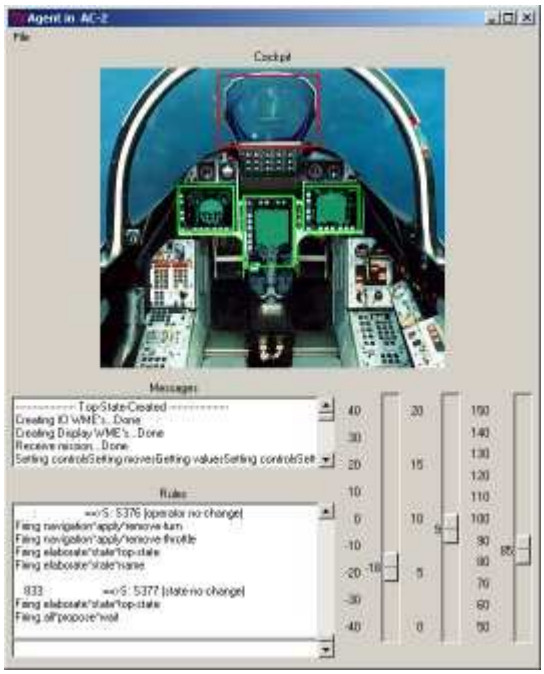
VVA-processen	?
Modellbyggarens gränssnitt	?
Återanvändbarhet/ utbyggbarhet	?
Integrerbart	?
Direkta prestationsmått	
Beräkningskapacitet	?
Övrigt	
Tillgänglighet	-
Hårdvara/mjukvara/OS	-
Huvudsaklig finansiär	-
Omfattning finansiering	-
Utvecklare	-
Användare	-
Användningsområde	-
Tillämpningar/exempel på projekt	-
Referenser	-

B. Framtagna Prototyper

Projektet har tagit fram några prototyper för att simulera datorgenererade styrkor. Prototyperna är alla modellerade för att simulera piloter, på grund av att projektet haft tillgång till experimentella data och anser sig mest hemma inom den domänen. Prototyperna har alla utnyttjat olika ramverk/simulatorer för att integrera datorgenererade styrkor. Detta beror på att man velat skapa erfarenheter från olika system för att kunna närma sig ”plug’n’play”- kompatibilitet.

Tcl-Pilot (2001)

Tcl-Pilot skapades under ett examensarbete som ingick i projektets förstudie. Simuleringsmiljön skapades i programspråket Tcl/Tk. Miljön bestod av en enkel karta och befolkades med flygplan och radarstationer. Till flygplanen kopplades två piloter som bl.a. använde Soar för beslutsfattning. En av piloterna hade som uppgift att flyga ett attackuppdrag, dvs flyga efter en brytpunktsbana för att sedan manövrera mot och attackera ett markmål. Den andra piloten flög ett patrullerande uppdrag, d.v.s. flyga efter en brytpunktsbana och jaga eventuella fiendeplan. Simulering resulterade i att jaktplanet började jaga attackplanet.



^

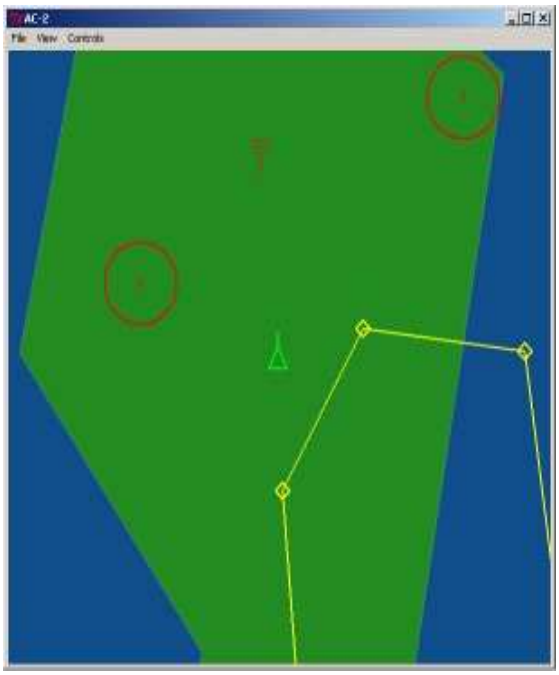
Det grafiska gränssnittet till cockpit i Tcl-Pilot. Cockpiten var uppdelad i fyra delar, där piloten kunde fokusera sin blick på en del i taget. I nedersta textrutan visas regler som exekveras av Soar-motorn. Till höger visas staplar som beskriver puls, upplevd höjd relativt riktig höjd och antal upplevda mål på radarn.

Det resonerande/planerande beteendet styrdes av arkitekturen Soar. Information om omvärlden skickades via simulerings-miljön, via ett ”cockpit” gränssnitt till Soar. Därefter exekverades regler i Soar beroende på omvärldsbeskrivningen. Några av dess regler skapade gärningar som t.ex. styrkommandon. I sista fasen skickades dessa gärningar via cockpit gränssnittet till simuleringsmiljön.

För att skapa ett något mer mänskligt beteende så skapades en modul för att styra blickriktning. Piloterna kunde således välja en av fyra displayer i cockpit att fokusera sina blickar på. Information skickades bara från den display som agenten valt att fokusera på. På så sätt var piloten tvungen att växla sin fokusering mellan de olika displayerna. När agenten tittade på en indikator uppdaterades arbetsminnet med den information som indikatorn visar. Agenten styrde flygplanen med en styrspak och navigerade genom att följa en simulerad kursmarkör.

En psykologimodul skapades för att simulera mental arbetsbelastning. Denna modul filtrerade information från agenten så att han vid stressade situationer inte exakt kunde avgöra vilken höjd han befann sig på och inte heller ange exakt antal radarupptäckta mål. Vidare förändras det visuella sökmönstret när agenten närmar sig målet, samtidigt som pulsen ökar markant. Det visuella sökmönstret, d.v.s. var agenten ”tittade” var uppdelat i flera faser med olika sökbeteende beroende på

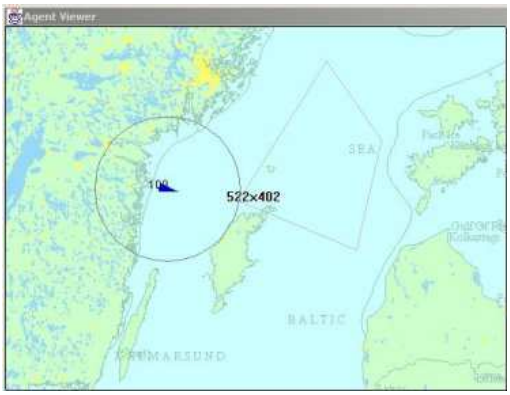
var i uppdraget agenten befann sig. Den visuella perceptionen styrdes av Markov modeller, d.v.s. en typ av finita tillstånds-maskiner. Fokuserings-mekanismen och psykologi-modulen byggde på data utifrån experiment med verkliga piloter.



< *Simuleringsmiljö för Tcl-Pilot. Grön triangel visar jaktplanet och röd triangel symboliserar attackplanet. Den gula polygonen visar en brytpunktsbana och de röda cirklarna symboliserar radar stationer och deras räckvidd.*

AF-Pilot (2001)

AF-pilot var en prototyp som utvecklades med en tidig version av *Agent Factory* från Pitch AB. Tillsammans med Pitch utvecklades en pilot med liknande beteende som Tcl-Pilot.



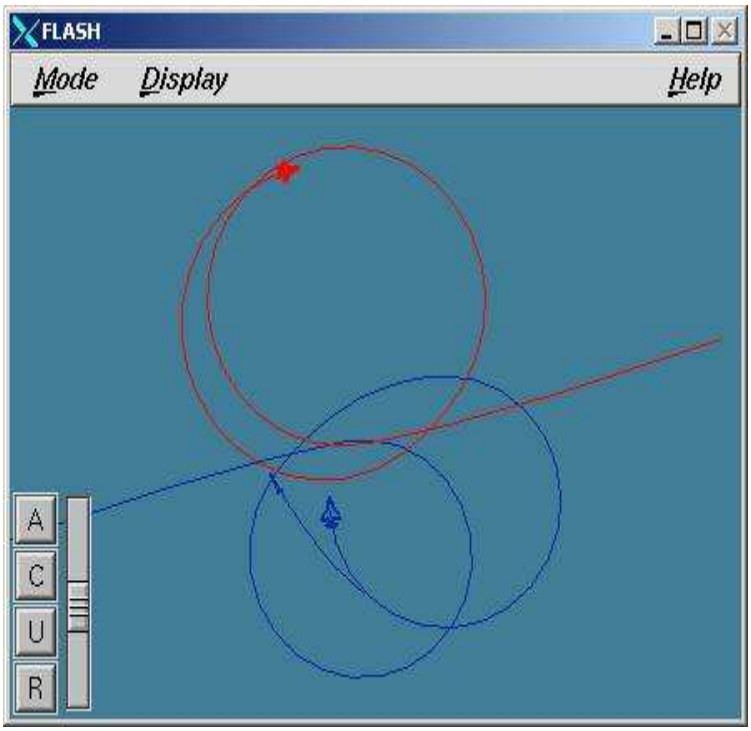
Utifrån Agent Factory skapades Java-kod som styrde agentens beteende. Agenten fick information och påverkade omvärlden via HLA. Visualiseringsmiljön som också lyssnade HLA kommunikation skapades av Pitch.

< *Visualiserng av en agent skapad i Agent Factory. Agenten var HLA-kompatibel.*

FLAMES-Pilot (2002)

FLAMES-Pilot skapades som ett försök att utnyttja gränssnitt från biblioteksstrukturen och en prototyp för att skapa ”plug’n’play” enheter. Prototypen är ett exempel på hur man på ett relativt enkelt sätt kan utnyttja datorgenererade styrkor i befintliga simuleringsramverk. FLAMES-pilot utnyttjade det gränssnitt, *AgentInterface* som projektet skapat till arkitekturen Soar. FLAMES är ett simulerings-ramverk som används inom FOI i flera sammanhang. I FLAMES kan man bl.a skapa egna entiteter och skapa egna scenarion. Här skapades bl.a ett scenario där två jaktflyg försöker att slå ut varandra. Jaktflygen skapades i FLAMES scenario verktyg, därefter kopplades de via FLAMES kognitiva processer och AgentInterface till Soar. Detta medförde att två Soar-agenter styrde varsitt jaktplan i scenariot.

Utifrån de gränssnittsbeskrivningar som ställts upp, krävdes det några få rader kod för att integrera Soar med Flames på sättet som beskrivs ovan. På grund av olika koordinatsystem var man tvungen att skapa nya vektor- och vinkelberäkningar. För att slippa sådana konverteringar kommer sådana typer av beräkningar att ingå som hjälpfunktioner i biblioteksstrukturen. Eftersom FLAMES inte har någon naturlig joystick var man tvungen att simulera en sådan, vilket också bör kunna ingå som hjälpfunktioner i biblioteket. Inga kognitiva begränsningar lades in i denna prototyp. I senare versioner bör man kunna styra andra entiteter än flygplan.



< *Skärmbild från FLAMES. I scenariot möts två jaktplan som styrs av Soar-agenter. Jaktplanen har som uppgift att slå ut varandra. På bilden ser man hur det blå planet just avfyrat en missil mot rött plan.*

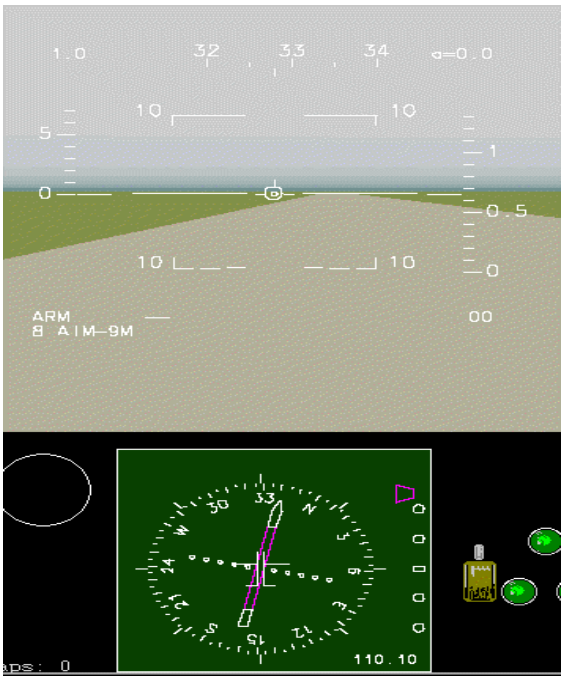
Acm-Agent (2002)

Acm är en flygsimulator som skapts i programspråket C. Simulatorn är DIS kompatibel, vilket gör att man kan köra flera flygplan på olika datorer. Källkoden finns tillgänglig för Unix, och har här modifierats något för att kunna köras under Linux.

Acm-Agent skapades för att kunna styra flygplan i Acm genom datorgenererade styrkor. Acm-Agent använde gränssnitt från projektets bibliotekstruktur för att integrera datorstyrt beteende via Soar och entiteter i Acm. Från applikationens sida, dvs Acm, implementerades det gränssnitt som skapats ur biblioteket. Denna implementation bestod av maximalt tre rader kod i C för varje funktion, vilket anses vara ett tecken på att gränssnittet är enkelt att implementera. Vidare användes prototypen Soar och *AgentInterface* för det resonerande/planerande beteendet. På samma sätt som i tidigare prototyper användes flödet ”input->reasoning->output”.

Agenten startade på marken sittandes i ett plan. Därefter frågade agenten ett simulerat flygtorn om tillåtelse att lyfta. Vid svar startade sedan agenten motorerna, drog upp till maximal motorstyrka. Vid bestämd hastighet drog sedan agenten joystick bakåt för att lyfta. Vid tillräcklig höjd, kommunicerade agenten till tornet att den nu var i luften. Därefter planade agenten ut beredd på fortsatt flygning.

Eftersom gränssnitts-anpassningen en gång är skapad, kan man därefter stoppa in olika agenter i simulatorn. Eftersom Acm har visat sig vara lätt att använda och lämplig att testa sina modeller i, kommer simulatorn även att användas vid senare studier. Mer komplexa beteende-modeller kommer att prövas, därtill samarbete och kommunikation mellan agenter. Acm lämpar sig bra eftersom dess funktionalitet även går att finna i många andra flygsimulatorer.



Skärmbild från Acm. I scenariot kommunicerar agenten med ett flygtorn för att simulera en start. Därefter flyger agenten fritt i miljön. >

Fenix-Pilot (Under utveckling)

Fenix är en flygsimulator som drivs av institutionen för Flyg och Autonoma System, FOI. Under året har en pc-baserad version utvecklats av institutionen. Under denna utveckling har ett samarbete inletts för att stoppa in datorgenererade styrkor i simulatorn.

^

Skärmbilder från Fenix.

v

Simulatorn stödjer även andra entiteter som markfordon och fartyg. En ny prototyp kommer därför att skapas där en agent kommer att styra en stridsvagn. I simulatorn kommer det att finnas funktionalitet för att se agentens styrspakslägen, blickriktning, m.m.